Personal Computer

MZ-2000

DISK BASIC MANUAL

SHARP

Personal Computer

MZ-2000

DISK BASIC Manual

ご　注　意

このマニュアルは、パーソナルコンピュータMZ-2000のシステムソフトウェアBASIC インタープリタMZ-2Z001に基づいて作成されています。

(1) 多目的パーソナルコンピュータMZ-2000では、システムソフトウェアをすべてファイル形態のソフトウェアパック（カセットテープ、ディスクなど）によってサポートされます。各システムソフトウェアおよび本書の内容は、改良のため変更することがありますので、ファイルバージョンナンバーには特にご注意されるよう、お願い致します。

(2) 本機は非常に複雑な機能および組合せを有する製品であり、出荷に際しては取扱説明書を含めて充分なチェックをして万全を期しておりますが、万一ご使用中ご不審な点、お気付きのことがありましたら、もよりのシャープのサービス窓口（技術サービス部・サービスステーション・サービスブランチ）までご連絡願います。なお、運用した結果生じる影響については責任を負いかねますのであらかじめご了承ください。

目　　次

は じ め に

このソフトウェアはディスクで供給しております。ディスクの取り扱いについては特に注意が必要ですので、取り扱い方法については本書121ページを参照ください。

ディスクオペレーション時にはマスターディスクを使用せず、マスターディスクをコピーしたサブマスターディスクを使用することを推奨します。不慮の事故によってマスターディスクが使用不能とならないように、安全な場所に保管ください。

（注意）　動作時のディスクの入れ換え

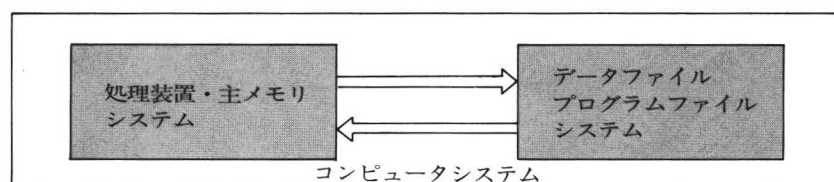
ディスクをフロッピードライブに挿入し、読み出し、書き込みを実行している時は、ディスクの入れ換えを行ってはいけません。

ディスクドライブ操作中にディスクを入れ換える必要がある場合には、ディスクの内容破壊防止のために、ディスクを入れ換える直前に、かならずKILL命令を実行してください。

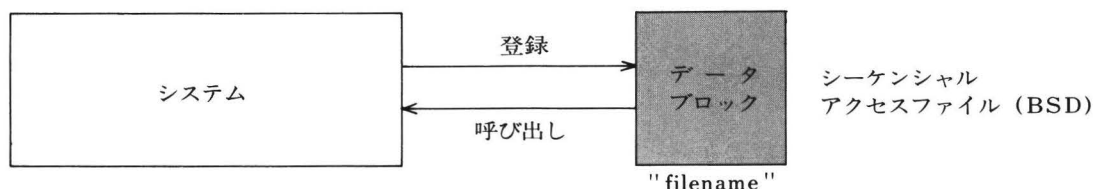
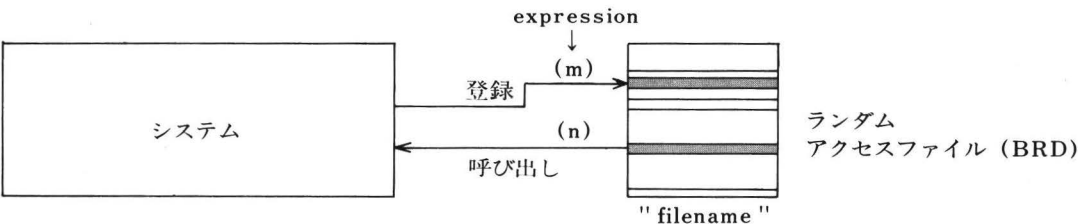
概　要　説　明

DISK BASICは、カセットベースBASICに対して、強力なファイルコントロール機能をもつ、ファイル指向のBASICとして、クリーンコンピュータシステムMZ-2000の新しいソフト領域を拡げるシステムソフトウェアです。すなわち、フロッピーディスクファイルの大容量で、しかも高速データアクセスが可能である特質を生かして、ファイルが単にデータの蓄積を荷うだけでなく、システムと直接に結びついたデータ・エリアとして使用することができるのです。さらに、CHAIN、SWAPというコマンドを用いて、プログラムテキスト自体を、ジョブ単位にオーバレイ、リンクさせて走らせることができ、プログラムの分割（セグメンテーション）という新しい手法を展開することができます。

また、このDISK BASICはGP-IBおよびRS-232Cインターフェイスに関するコントロールを含んでいますので、GP-IBインターフェイスカードMZ-8BI04、シリアルインターフェイスカードMZ-8BI03を用いて、コンピュータMZ-2000により機器をコントロールする場合、このBASICを用いることができます。

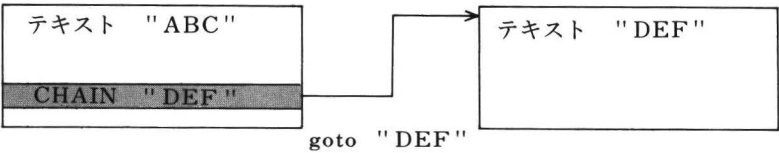
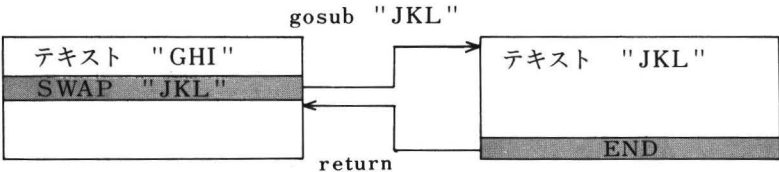
DISK BASICを理解され、その機能を充分発揮されることによって、スモールビジネスや研究室での本格的な、より高度なシステム活用が可能になるものと考えられます。コンピュータシステムは、そのデータ処理装置と主メモリとが構成するロジカル的な内部システムと、そこで処理されるデータやプログラムバンクが構成する外部ファイルシステムの、2つの大きなシステムに分けて考えることができますが、フロッピーディスク装置およびDISK BASICは、このファイルシステムコントロールを司る柱としての重要な役割を果すことになるからです。

データファイルコントロール

データファイルには、ファイルアクセスの形式に従って、2種類あり、1つはシーケンシャルアクセスファイル、もう1つはランダムアクセスファイルと呼ばれます。

シーケンシャルアクセスファイルとは、ファイルデータのアクセスを、ひとつながりのブロックとして扱うもので、一連のデータ群を、ファイル名を指定して、ファイル上に登録、あるいは呼び出す際、各データは、その並びの先頭から順次アクセスされます。

それに対して、ランダムアクセスファイルでは、ファイル中のデータアクセスが、ランダム形式で行われます。すなわち、1つのランダムアクセスファイルは、1つのファイル名で指定されるデータ群を構成しますが、その個々のデータは、ファイル上に1次元配列要素の形で登録されており、各データの書き込み、読み出しは、そのデータ要素番号(expression)を指示することによって行われます。

一般に、あるデータの集合が、一連のものとして捉えられるとき(たとえば、POKE文で機械語プログラムを作成するとき使う、10進表現の一連のデータや、先頭から順次並べられた、表の要素など)、シーケンシャルアクセスファイル上にそのデータ集合を登録するのが有効だといえます。また、データの集合が全体としてだけでなく、個々の要素としてアクセスされる必要のあるとき、たとえば、データの書き替え、検索、並べ替え、削除などが必要であるとき、ランダムアクセスファイル上のデータとして構成するのが有効だと言えます。

プログラムファイルコントロール

プログラムファイルコントロールコマンド、CHAINおよびSWAPは、プログラム実行中に、他のプログラムファイルをオーバレイして、コントロールを移すためのものです。

CHAINは、下図のように、「goto "filename"」としての機能を持っています。

SWAPは、「gosub "filename"」としての機能を持ち、オーバレイサブプログラムの終了後、はじめのプログラムに戻る(この時もオーバレイが行われます)ことができます。

この他に、機械語プログラムファイルをロジカルオープンしてリンクするコマンドもあります。また、ユーティリティプログラムを用いた各種のファイルマネージメントを行うことができます。

DISK BASIC 概要

Chapter 1

この章は、DISK BASICインタープリタMZ-2Z001の特徴、ファイルコントロールの概要について解説を行っています。はじめに、DISK BASICの起動方法が説明されます。

DISK BASICの持つ新規コマンド、ステートメントの文法解説は第２章に、DISK BASICの持つ全てのコマンド、ステートメント、関数、オペレータのまとめは第4章です。

1.1 DISK BASIC インタープリタMZ-2Z001の起動

DISK BASICインタープリタMZ-2Z001（およびMONITOR MZ-1Z001M）はマスターディスク中にあるので、それを走らせるには先ず、IPLによってイニシャル、ローディングを行わなければなりません。

イニシャル・ローディングは簡単に実行することができます。MZ-2000（電源OFF）にフロッピーディスクドライブを接続した状態で、ドライブの電源をONにし、ドライブ1番にマスターディスクをセットします。そして、MZ-2000の電源をONすると、DISK BASICが自動的に起動されます。†）

図1－1の左側は、DISK BASICをローディング中であることのメッセージ表示を、右側は、ローディングを終了して、DISK BASICインタープリタMZ-2Z001（およびMONITOR MZ-1Z001M）が起動し、カーソル点滅によってBASICコマンドレベルになったことを、それぞれ示しています。

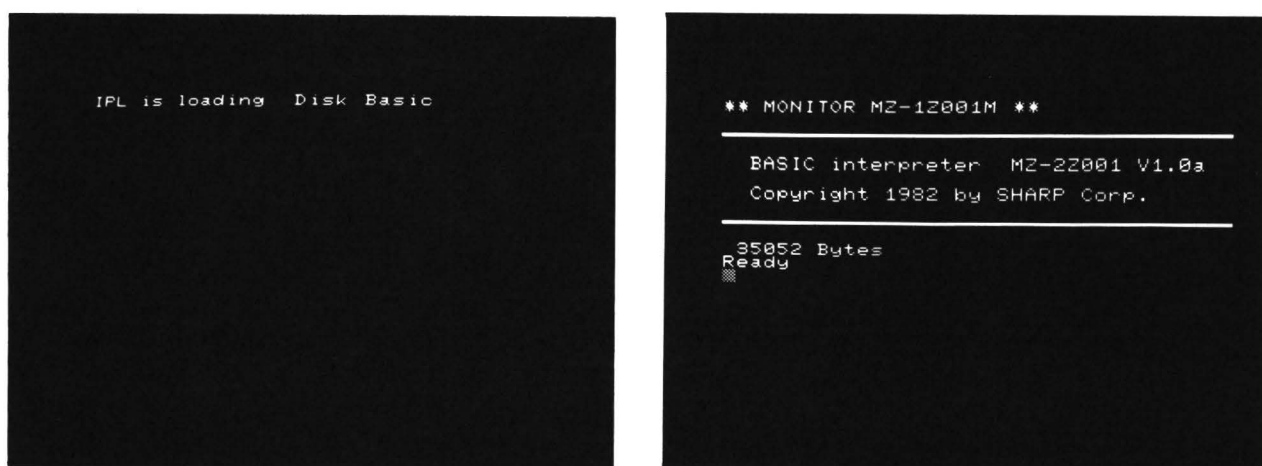



図 1—1

BASICテキスト"AUTO RUN"が自動的に実行されること

上記のオペレーションのうちに含まれている、BTX"AUTO RUN"の実行を解説します。DISK BASICがローディングされ、テキストエリアのバイトサイズ（上例では35052バイト）が表示されると、再び、マスターディスクがアクセスされるのに気が付かれたと思いますが、DISK BASICは、イニシャルローディングを終えた時、自動的に、

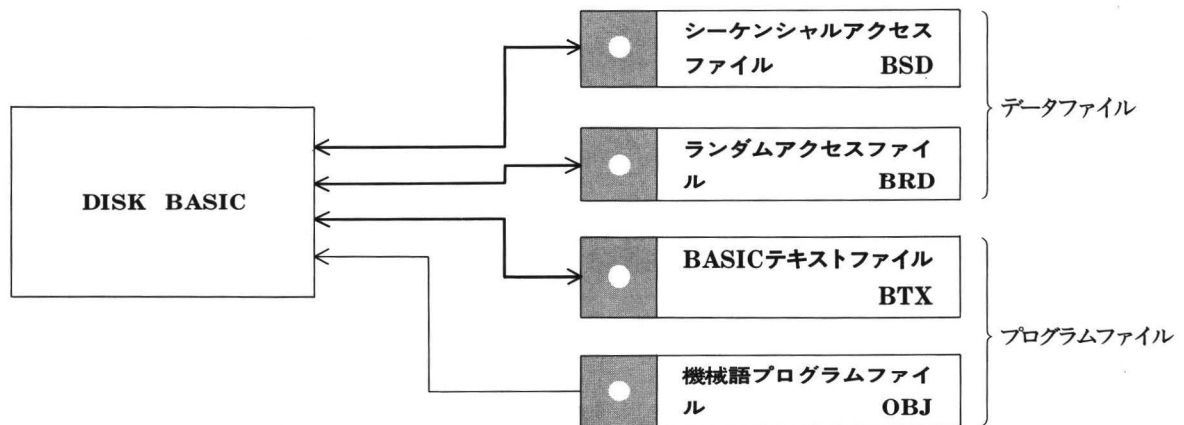
RUN "AUTO RUN"

コマンドを実行します。即ち、"AUTO RUN"というファイル名のプログラムテキストを同じマスターディスクからロードし、その先頭から実行します。サポートされたマスターディスク上には、デファイナブルファンクションキーを定義するプログラムをこのファイル名で登録しています。また、このプログラムの最後に、NEW文を置いているので実行後、"Ready"を表示しカーソル点滅となる前にテキストは消去されているのです。（一度、LOAD"AUTO RUN"を行ってリストを調べてみて下さい。）

もし、DISK BASICの起動に続けて、或るプログラムをスタートさせたい場合、そのプログラムのファイル名を"AUTO RUN"としてマスターディスク上にセーブしておけばよいことになります。

†）ファイルのイニシャルローディングは、カセットファイルよりも、ディスクファイルが優先となっており、またディスクドライブは1番が自動的にロード・アクセスされるのでこの自動オペレーションが行われます。ドライブ2番によってイニシャル・ローディングを行う場合等の操作方法は、Owner's Manual 第2章に解説されています。

1.2 ファイルコントロール

DISK BASICで作成されるファイルは、シーケンシャルアクセスファイル(BSD)とランダムアクセスファイル（BRD）の2種類のデータファイルと、BASICテキスト（BTX）のプログラムファイルの3種類があることを、ファイルの説明の項で述べました。もう一種類の機械語プログラムファイル（OBJ）は、MONITORプログラムあるいはFDOSなどで作成したものをディスク上へ登録したもので、それは、単独で走らせるか、BASICの機械語エリアへ乗せて、BASICテキストとリンクして使うなどの目的をもったファイルです。したがって、DISK BASICで利用することはできても、DISK BASICで作成したり内容を変更したりするファイルではありません。

各種ファイルコントロールコマンドの解説にあたって、はじめに、2種類のデータファイルの作成方法、利用方法、それぞれの特徴を説明し、つづいて、プログラムファイルのCHAIN、SWAPコマンドの使い方を説明して行きます。

1.3 シーケンシャルアクセスファイルのコントロール

シーケンシャルアクセスファイルとは、データの登録または呼び出しが、シーケンシャルアクセス形式で行われるデータファイルのことです。**シーケンシャルアクセス(sequential access)** 形式とは、データのアクセスを、先頭から順番に行う形式を意味するものです。

すでに、MZ-1Z001シリーズのBASICで、カセットファイル上へ、データファイルを作成する方法を説明していますが、DISK BASICでのシーケーンシャルアクセスは、ちょうどそれと同じことを、ディスクに対して行うものです。もちろん、はるかに高速アクセスができ、さらに幾つかの新しいファイルコントロールコマンドを用いることによってファイル管理の面でも幅のある使用が可能となります。

最初に、DISK BASIC と、カセットベースのBASIC のそれぞれのシーケンシャルアクセスコマンドの構成を、対比させてみることにします。

ファイルの登録（データの書き込み）

	DISK BASIC	カセットベースBASIC
ファイルオープンコマンド	WOPEN　#*n*, "*filename*"	WOPEN　"*filename*"
データ書き込みコマンド	PRINT　#*n*, *data*	PRINT／T　*data*
ファイルクローズコマンド	CLOSE　#*n*	CLOSE
キャンセルコマンド	KILL　#*n*	———

ファイルの呼び出し（データの読み出し）

	DISK BASIC	カセットベースBASIC
ファイルオープンコマンド	ROPEN　#*n*, "*filename*"	ROPEN　"*filename*"
データ読み出しコマンド	INPUT　#*n*, *variable*	INPUT／T　*variable*
ファイルクローズコマンド	CLOSE　#*n*	CLOSE
ファイルエンドの検出	IF EOF (#*n*) THEN	———

注）オープンコマンドの一般形は、ドライブ番号とディスクボリュームナンバの指定を含みますが、上の2つの表では省略形で示しています。

それぞれのコマンドの構成を比べてみて、殆ど、一対一に対応していることがわかると思いますが、DISK BASICのコマンドにはいつも、#*n*という要素が含まれていますがこれは、**ロジカルナンバ(logical number)** と呼ばれる番号で、DISK BASICのファイルアクセスは、常にこの番号を指定して行わなければなりません。

カセットベースBASICでは、ファイルアクセスはデータの書き込みか読み出しかを1つのファイルについて実行するだけしかできませんが、DISK BASICでは、ディスクという、ファイルが複数並置され任意にアクセスできるシステムを活かして、同時に複数個（**最大10個**）のファイルをコントロールすることができます。そして、ファイルをオープンしたら、それぞれ任意に選んだロジカル番号に定義しておき、それ以後該当ファイルの指定は、ロジカルナンバを使って、わざわざファイル名を書かなくて済むようにしているのです。

簡単な例として、人名とその自宅の住所とをシーケンシャルアクセスファイナルに登録することを考えてみます。手元にある住所録、たとえばあなたが同窓会の幹事をしているとしたらそのクラス全員の名簿を片っ端から全部ファイルにとっておくのです。たとえば、次のようなファイルとして……。

filename＝"3－H、クラスメイボ"

人名も住所も長さをバラバラに書いたのは、**一般にシーケンシャルアクセスで登録されるデータは固定長ではなく、データによって長さが異る**からです。後で出て来るランダムアクセスファイルではデータは全て32バイトの固定長の箱に収められます。今の場合のように、データが、１まとまりのものとして扱われ（３年H組という１クラス全体)、また住所のようにたいてい32バイトでは足りなく、長さもまちまちの場合、シーケンシャルアクセスファイルに登録するのが向いています。

INPUT文で、人名と住所とを交互にストリング変数に代入して、１人ずつファイルに登録して行き、全部で50人分の住所録"３－H、クラスメイボ"を作成し、次に作成されたファイルを読み出して、10人分ずつ名前と住所とをCRTディスプレイに表示していくプログラムは、次のようにして作ることができます。

〔書き込み〕

```
100 WOPEN #3,FD1@22,"3-H,クラスメイボ"
110 FOR P=1 TO 50
120 INPUT "ナマエ=";NA$
130 INPUT "ジュウショ=";AD$
140 PRINT #3,NA$,AD$
150 NEXT P
160 CLOSE #3
```

〔読み出し〕

```
200 ROPEN #4,FD1@22,"3-H,クラスメイボ"
210 FOR P=1 TO 5 :FOR Q=1 TO 10
220 INPUT #4,NA$,AD$
230 PRINT NA$ :PRINT AD$
240 NEXT Q
250 PRINT "STRIKE ANYKEY"
260 GET X$:IF X$= ""THEN 260
270 NEXT P
280 PRINT "END"
290 CLOSE #4
```

■ファイルエンドのみつけ方

ファイルからデータを順番に読み出して行って、登録されているデータを越えた場合にどうなるか？……この場合は、エラーは発生せず、読み出す変数には0か空がセットされることになるのですが、特別な関数、EOF（#n）というものがあり、これがファイルエンドを捉えます。**EOF(#n)は、INPUT#コマンドでデータの読み出しを行った時、そのファイルエンドに来ていたら条件が真となります。**したがって、INPUT#文の次に、

IF EOF (#n) THEN

コマンドを置いておくと、EOF（#n）が"真"、すなわちファイルエンドを見つけ出した場合に、THEN以下が実行されることになります。

〔問　題〕 前ページのプログラム例で、"3—H、クラスメイボ"の読み出しで、これに登録されている人数が不明だとして、10人分ずつファイルエンドまで読み出し、表示するプログラムに変更しなさい。

〔例　解〕 たとえば次のようなプログラムが考えられます。

```
300 ROPEN #5, FD1@22, "3—H, クラスメイボ"
310 FOR I=1 TO 10
320 INPUT #5, NA$, AD$
330 IF EOF (#5) THEN 400
340 PRINT NA$:PRINT AD$
350 NEXT I
360 PRINT "STRIKE ANYKEY"
370 GET X$:IF X$=" "THEN 370
380 GOTO 310
400 CLOSE #5
410 PRINT "FILE END":END
```

練習問題

〔問　題〕 BSDファイル"3—H、クラスメイボ"を人名だけ登録したBSDファイルと住所だけ登録したBSDファイルの2つに分けて登録し直してみよ。

〔例　解〕

```
500 ROPEN #6, FD1@22, "3—H, クラスメイボ"
510 WOPEN #7, FD1@22, "ナマエ"
520 WOPEN #8, FD1@22, "ジュウショ"
530 INPUT #6, NA$, AD$
540 IF EOF (#6) THEN 600
550 PRINT #7, NA$
560 PRINT #8, AD$
570 GOTO 530
600 CLOSE
610 END
```

〔問　題〕 INPUT文でキー入力したストリングをBSDファイルに登録する。ただし、ファイルをクローズする場合"CLOSE"を、キャンセルする場合"KILL"をキー入力するものとします。

〔例　解〕

```
100 WOPEN #30, "SEQ DATAS"
110 INPUT "DATA="; A$
120 IF A$="CLOSE" THEN CLOSE #30:END
130 IF A$="KILL" THEN KILL #30:END
140 PRINT #30, A$:GOTO 110
```

1.4 ランダムアクセスファイルのコントロール

ランダムアクセスファイルとは、データの登録または呼び出しが、ランダムアクセス形式で行われるデータファイルのことです。**ランダムアクセス(random access)**形式とは、ファイル中のデータのアクセスを、配列形式として指定することによって行うということです。つまり、シーケンシャルアクセスの場合は必ず、データ集合の先頭から順番にアクセスが行われるのに対して、ランダムアクセスファイルでは、データ集中の任意のデータ要素をいきなりアクセスすることができるのです。

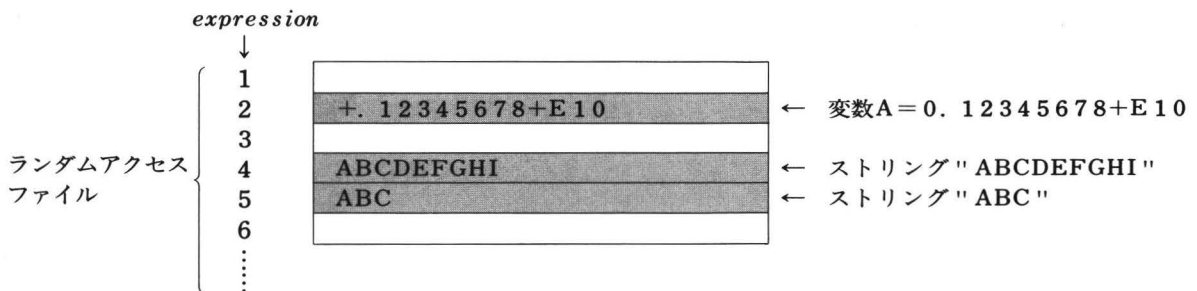
ランダムアクセスファイルを構成しているデータ集合に対して、その配列要素を指定してデータアクセスを行うために、ランダムアクセスコマンドでのPRINT＃文、INPUT＃文は、次のように、ロジカルナンバにつづいて、*expression*（配列要素指定式）が置かれます。

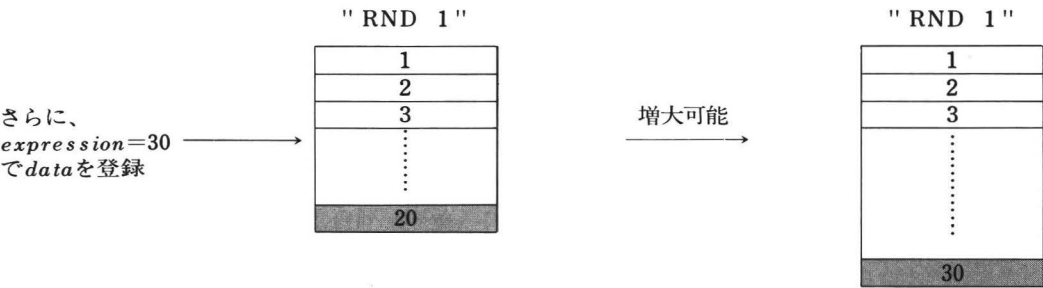
PRINT ＃*n*(***expression***)、*data*
INPUT ＃*n*(***expression***)、*variable*
↑
配列要素の指定

expression は数値または変数で与えます。たとえば、

INPUT ＃7 (21)、A＄

というステートメントは、ロジカルナンバ＃7にオープンされているランダムアクセスファイルのデータ集合中で、21番目の要素として登録されているデータを、ストリング変数A＄に読み出せという意味になります。

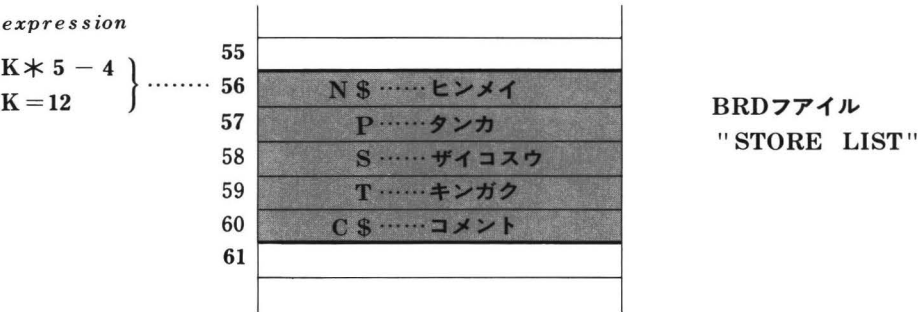
ランダムアクセスファイルのこのようなデータアクセスを可能にしている条件として、各データは、固定長で登録されるということに注意しなくてはなりません。即ち、**数値およびストリング変数はファイル上に登録されるとき、いずれも32バイトで区切られた"箱"の中にセットされる**のです。

数値変数の場合は、指数表現の場合でもいつも32バイト内におさまりますが、ストリング変数の内容は、最大255バイトの長さまで持つことができるので、32バイトを越えるストリングは、ランダムアクセスファイルの1つのデータ要素に登録することができないことになります。

シーケンシャルアクセスファイルと異るもう一つの点は、ファイルが一旦クローズされ登録された状態になっていても、同じファイルをさらに大きくして行くことができることです。たとえば、*expression* が20まで使われて登録されたランダムアクセスファイル"RND 1"があったとしたとき、新たに *expression* を30としてデータを登録すると"RND 1"は、30の"箱"を持ったファイルに増大して行きます。

今、ランダムアクセスファイルを利用して、簡単な在庫リストファイルを作成することを考えてみましょう。それぞれの品物には、No.1からNo.50までの固定した品番があるものとして、ファイルには、品名、単価、在庫数、金額(単価×在庫数)、コメントの4項目を置くことにします。

それぞれの品物について、在庫データを登録するのに、はじめに品番を入力しておいて、次に各項目に登録する内容を入力するようにします。

在庫データの登録

```
100 XOPEN #5, "STORE LIST"
110 INPUT "ヒンバン=";K
120 IF K=0 THEN 300
130 INPUT "ヒンメイ=";N$
140 INPUT "タンカ =";P
150 INPUT "ザイコスウ=";S
160 INPUT "コメント=";C$
170 T=P*S
180 PRINT #5(K*5-4),N$,P,S,T,C$
190 GOTO 110
300 CLOSE #5
310 END
```

これによって作成されるランダムアクセスファイルは、次のようなものになります。品番をK=12とすると、配列引数(*expression*)が56〜60に相当する要素に5のデータが登録されます。

このように、データはファイル上の任意に指定できる配列にセットすることができます。したがって、先頭から順番にデータがつめられたシーケンシャルアクセスファイルと異って、ファイル中にデータの空番地ができることもあります。またデータの書き換えも簡単にできることになります。

次に、ここで作成したランダムアクセスファイル"STORE LIST"を呼び出して、ある品物の在庫データを読み出すことを考えてみましょう。

在庫データの呼び出し

```
500 XOPEN #17,"STORE LIST"
510 INPUT "ヒンバン=";J:IF J=0 THEN 700
520 INPUT #17(J*5-4),N$,P,S,T,C$
530 PRINT "NO.";J:PRINT "ヒンメイ  ";N$
540 PRINT "タンカ    ";P
550 PRINT "ザイコスウ";S
560 PRINT "キンガク  ";T
570 PRINT "コメント  ";C$
580 GOTO 510
700 CLOSE #17
710 END
```

このように、品物が多数あっても所望の品物の品番を入力することによって直ちにその在庫データを呼び出すことができます。

1.5　プログラムのCHAIN

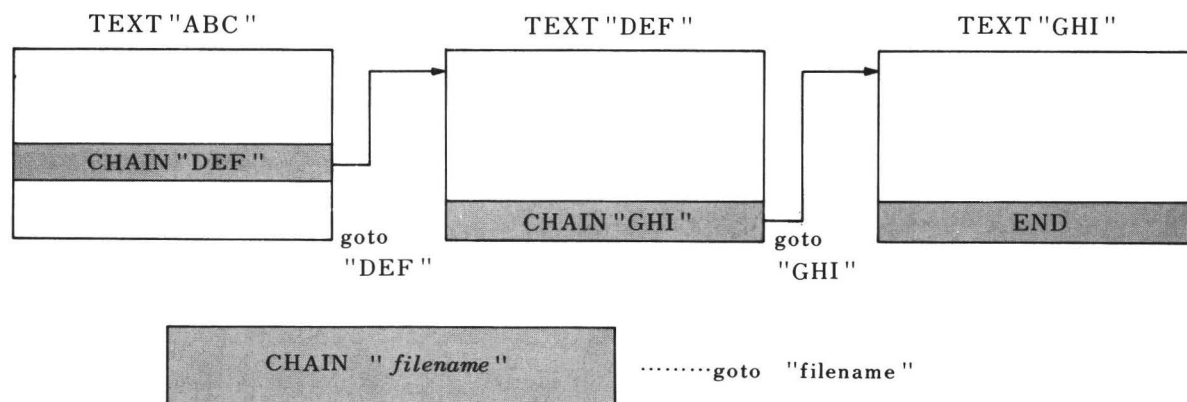
データファイルコントロール命令に続いて、プログラムファイルのコントロールについて説明いたします。ここで説明される命令は、CHAINとSWAPの2つのコマンドです。このコマンドを用いると、プログラムをジョブ単位でディスク上に登録しておいて、プログラムを走らせながら、別のプログラムを呼び出してそれにコントロールを移すことができます。つまり、プログラムを、ディスク上に登録されているプログラムに接続したり(CHAIN)、サブルーチンの形で呼び出したり（SWAP）することができるのです。はじめに、プログラムを接続、連結するCHAINコマンドから説明いたします。

CHAIN コマンドの形式はたとえば次のようになっています。

```
700 CHAIN FD1@50,"TEXT 2"
```

このステートメントは、現在テキストエリア内にあるプログラムをNEW して(ただし変数の値は保持する)、ドライブ1番中のボリュームナンバ50のディスク上に登録されている、"TEXT　2"というファイル名のテキストを、オーバレイ（テキストエリアへ重ねて読み出すこと）してそのテキストの先頭へコントロールを移しなさい、という意味です。

このステートメントを実行させると、現在走っているBASICテキストからコントロールを離れて、新たにテキスト"TEXT　2"を読み出して来てその先頭へコントロールが移ります。**プログラムのCHAINが実行される時、変数の値と、DEF　FN で定義した関数とは、CHAIN 先のプログラムへ受渡されます。**

CHAIN コマンドの機能は、「goto　"filename"」として捉えることができます。

CHAINコマンドを使うと、大きなプログラム――BASICテキストエリアを越えてしまうような巨大なプログラムでも、それを分割しておいて上のように継ぐことができます。1つのプログラムを終了したら、データをそのままにしておいて、次のプログラムに次々にCHAIN して行くのです。たとえば、スモールビジネスでの複雑で多岐にわたるデータ処理を行わなくてはならない場合などでは、CHAINや、この次に出て来るSWAPコマンドは、なくてはならないものといえます。

1.6 プログラムのSWAP

SWAPコマンドは、ディスクファイル上のプログラムを呼び出して、オーバレイ、リンクを行いそのテキストにコントロールを移しますが、そのプログラムを終了したら、もとのプログラム（SWAPコマンドを行ったプログラム）に復帰することができます。この動きは、ちょうどテキスト中のサブルーチン参照と同様であり、もとのプログラムへの復帰は、SWAPコマンドを行った次のコマンドへ戻ることになります。したがってSWAPコマンドを、「**gosub"filename"**」として捉えることができます。

この動作を行うのに、SWAPコマンドをもつプログラムテキストは、SWAP実行時に一時ディスク上へ待避されなくてはなりません。そのうえで、テキストエリアがNEWされ、サブプログラムの呼び出し実行が行われ、サブプログラムの終了後に、待避していた元のプログラムテキストへ復帰が可能になるのです。SWAPコマンドの一般形は次のようになっています。

SWAP FD*d*@*v*, "*filename*"

即ち、ドライブ*d*番（*d*＝1～4）中の、ボリュームナンバ*v*番のディスク上に登録されている、"*filename*"で指定するサブプログラムをSWAPせよ、という意味ですが、サブプログラムの待避に先がけて行われるプログラムテキストの待避は、一番最近DIRコマンドを実行しているドライブ中のディスク上へ行われます。従って、そのドライブ中には、プログラムテキストの一時書き込みが可能なディスクがセットされていなければならないことになります。SWAPレベルは、1レベルを超えてはなりません。

SWAPコマンドを理解するために、簡単な例をとってプログラムファイルの動きを追ってみることにしましょう。DIR 1コマンドを実行しているときを考えてみます。

〔現在テキストエリアにあるプログラム〕

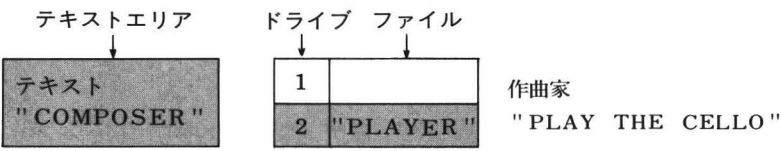
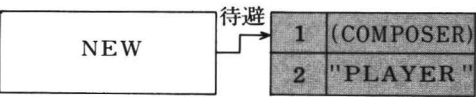
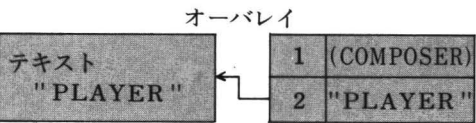
```
10  REM COMPOSER
20  M1$="A7B6+C3A7A3"
30  M2$="B+C+D+E6A3"
40. M3$="+F6A3+E7"
50  PRINT "PLAY THE CELLO"
60  SWAP FD2 @7 ,"PLAYER"
70  PRINT "VERY GOOD"
80  END
```

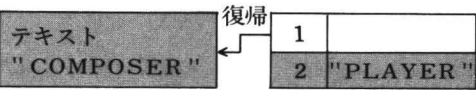
〔プログラムファイル"PLAYER"〕

```
10  REM CELLO PLAYER
20  MUSIC M1$,M2$,M3$
30  PRINT"OK?"
40  END
```

↑
ドライブ2番中のボリューム7番のスレーブディスク上にあります。

はじめ、"COMPOSER"がテキストエリア内にあって、実行している。

作曲家
"PLAY THE CELLO"

行番号60のSWAPコマンドで、まず、テキストが、DIRをとっているＦＤ１中のディスクに待避され、エリアがNEWされる。

つづいてBTX"PLAYER"がオーバレイ実行される。メロディーが演奏される。

演奏家、メロディーをひく

終わると、待避していたCOMPOSERが復帰して"VERY GOOD"といいます。

作曲家
"VERY GOOD"

1.7 予約語

BASIC文は予約語（reserved words）——キーワードとも呼ばれます——と、そのオペランド、セパレータ、データによって構成されます。予約語は、BASICインタープリタがそれを解釈し決められた機能を行う特別な語であり、コマンド、ステートメント、関数がそれに相当します。予約語は、特殊な機能を実行させるために決められている語なので、プログラマが、変数名、配列名などに用いることができません。表1－1は、DISK BASIC MZ-2Z001の全ての予約語をアルファベット順に並べて示しています。(表１－１の予約語の右の数字は参照ページを示しています。)

A	ABS	87		ERL	40		MON	77		ROPEN/T	80
	APPEND	29		ERN	39		MUSIC	83		RSI	113
	APPEND/T	29		ERROR	39	N	NEW	77		RSMODE	112
	ASC	88		EXP	87		NEXT	81		RSO	113
	ATN	87	F	FAST	83	O	ON	105		RUN	30
	AUTO	77		FOR	81		OUT	87	S	SAVE	28
B	BLINE	84	G	GET	81	P	PAGE/P	86		SAVE/T	28
	BOOT	77		GOSUB	81		PATTERN	84		SET	84
C	CHAIN	33		GOTO	81		PCT	98		SGN	87
	CHANGE	83		GPIBM	108		PEEK	85		SIN	87
	CHARACTER$	88		GRAPH	84		POINT	85		SIZE	83
	CHR$	88	I	ICL	94		POKE	85		SPACE$	88
	CLOSE	35		IF	82		POSH	85		SPOL	105
	CLOSE#	35		IMAGE/P	86		POSITION	84		SQR	87
	CLOSE/T	80		INP	87		POSV	85		SRQ	105
	CLR	83		INPUT	80		PPC	106		STEP	81
	CMDR	104		INPUT#	36		PPOL	107		STOP	82
	CMDW	103		INPUT/T	80		PPU	108		STR$	88
	CONSOLE	83		INT	87		PRINT	80		STRING$	88
	CONT	77	K	KILL	35		PRINT#	34		SWAP	33
	COPY/P	86		KLIST	77		PRINT/P	86	T	TAB	89
	COS	87	L	LCL	95		PRINT/T	80		TAN	87
	CSRH	83		LEFT$	88	R	READ	81		TEMPO	83
	CSRV	83		LEN	88		RED	100		THEN	82
	CURSOR	83		LET	80		RED/	102		TI$	83
D	DATA	81		LIMIT	85		REM	82		TO	81
	DCL	96		LINE	84		REN	94		TRG	97
	DEF FN	82		LIST	77		RENAME	32	U	UNLOCK	31
	DEF KEY	82		LIST/P	77		RESET	84		USR	86
	DELETE	32		LLO	96		RESTORE	81	V	VAL	88
	DIM	82		LN	88		RESUME	40		VERIFY	77
	DIR	26		LOAD	29		RETURN	81	W	WOPEN#	34
	DIR/P	27		LOAD/T	29		REW	83		WOPEN/T	80
E	END	82		LOCK	31		RIGHT$	88		WRT	99
	EOIR	108		LOG	87		RND	88		WRT/	101
	EOIW	108	M	MID$	88		ROPEN#	36	X	XOPEN#	37

表 1−1 DISK BASICインタープリタMZ-2Z001の全ての予約語

1.8　初期設定値について

DISK BASICインタープリタMZ-2Z001がIPLによって起動した時の、システム変数等のデフォルト値はそれぞれ次のように設定されます。

■　キーボード関係

1)　動作モード：ノーマルモード

2)　小文字の入力はノーマルモードのシフトポジション

3)　デファイナブルファンクションキーは、BTX " AUTO　RUN " によって次のように設定されます。

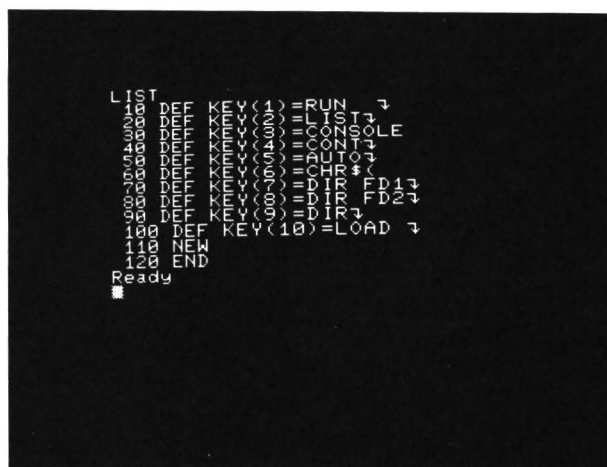



図　1−3

■CRTディスプレイ関係

1)　キャラクタディスプレイモード：　ノーマル（バックグラウンド：黒）

2)　キャラクタ表示桁数：　40キャラクタ／行

3)　キャラクタ表示スクローリングエリア：　最大（第0行から第24行）

4)　グラフィックディスプレイ入力モードページ：　ページ1
　　グラフィックディスプレイ出力モードページ：　全ページともOFF
　　ポジションポインタ：　POSH＝0、POSV＝0

5)　グラフィックディスプレイリゾリューションモード：320×200ドット／画面

■内蔵時計

TI＄＝" 000000 " で初期化してスタートする。

■音楽機能

1)　テンポのデフォルト値：　4（中庸のテンポ、Moderato）

2)　音長のデフォルト値　：　5（4分音符、♩）

■その他

1)　配列変数は全て未定義

2)　BASICテキストエリアの上限：　＄FFFF番地（即ちLIMIT　MAX　状態）

3)　タブセット：5キャラクタ毎

DISK BASIC MZ-2Z001の新規コマンドとステートメント

Chapter 2

この章は、DISK BASIC MZ-2Z001について、BASIC MZ-1Z001にない新規コマンドとステートメントの文法を解説します。

コマンドとステートメントの書式

■コマンドおよびステートメントを、アルファベット小文字、反転文字で表記することはできません。

■オペランド中でプログラマが指定すべき箇所は、イタリック体で示しています。

■かぎカッコ 〈 〉 中の要素は、省略あるいは、任意回の繰り返し記述が可能な要素であることを示しています。

■セパレータ（コロン、セミコロンなど）は、決められた位置に正しく置かなくてはなりません。

2.1 コマンド

2.1.1 DIR

書　式　DIR <FD *d*>

d………ドライブ番号：*d*＝1～4

機　能　FD *d* で指定するドライブ中のディスクのディレクトリをCRTディスプレイ上に表示します。
(DIR：directory)

解　説　ディレクトリとは、該当ディスクに既に登録されているファイルの情報を得るものであり、次の各データが表示されます。

〔1〕　ディスクのボリュームナンバ

スレーブディスクについてはそのボリュームナンバ（イニシャライズ時に決めたディスク番号）を、マスターディスクであれば"MASTER"を表示します。

〔2〕　空きセクタの総数

ファイルを1つも登録していないスレーブディスクは空きセクタ数は1072セクタです。

〔3〕登録されている全てのファイル

各ファイルについて、ファイルモード、ロック／アンロック、ファイル名を示します。
ファイルモードは4種類あり、次の各3文字のコードによって区別されます。

BTX………BASICテキストファイル
BSD………BASICシーケンシャルアクセス・データファイル
BRD………BASICランダムアクセス・データファイル
OBJ ………オブジェクトファイル

ロック／アンロックは、アスタリスク"＊"によって区別されます。即ち、ファイルモードに続けて"＊"の表示されているファイルはロックされた（固定された）ファイルです。
ロックされたファイルは、その状態ではデータの追加、ファイルの削除、ファイル名の変更が禁止されます。
ファイル名は、ファイル登録時に指定した名前で、ファイルの呼び出しはいつもその名前によって行います。
ディレクトリ表示の順序は必ずしもファイル登録順になるとは限りません。

ディレクトリ表示は、ファイルを1画面ぶん（20個のファイル）表示すると、そこで一旦停止して、カーソルが現れます。更に続けてディレクトリ表示を行わせるには［CR］キーを押しますが、その状態から他のコマンドへ移ることもできます。
LOADコマンド、RUNコマンドの実行は、ディレクトリ表示上でカーソルを該当ファイルの行へ移動させ、ファンクションキー(BTX　"AUTO　RUN"によってLOADコマンドは［F10］にRUNコマンドは［F1］に定義されています）を押すことによって簡単に行うことができます。
なお、このDIR命令を実行すると、グラフィックディスプレイ出力モードページは、全ページともOFFとなります。(DIR／Pも同様)

例　図2－1は、DIR FD1コマンドによって、マスターディスクのディレクトリを表示させた1例です。(実際のマスターディスクの内容は、この表示内容と若干異なる場合があります。)

図 2—1

2.1.2 DIR/P

書　式　DIR 〈FD *d*〉/P.

d………ドライブナンバ：*d*＝1～4

機　能　ディスクのディレクトリをラインプリタ上へプリントアウトさせます。この場合、CRTディスプレイ上への表示は行われません。

2.1.3 SAVE

書　式　SAVE 〈FD d @ v ,〉 "*file name*"

d………ドライブナンバ： $d = 1 \sim 4$

v………スレーブディスク・ボリュームナンバ

機　能　現在BASICテキストエリアにあるプログラムテキストを、ファイル名を指定してディスク上へセーブします。

解　説　SAVEコマンドを実行することによってディスク上に1つのBASICテキストファイル（BTX）が登録されます。その際、オペランドに必ずファイル名となるストリングデータを記述しなくてはならず、クォーテーションマークで囲った16文字以内のストリングで指定します。

オペランドFD d @ v はデフォルト可能な要素であり、省略された場合は、デフォルトドライブ(即ち、最後にDIR FD d コマンドによって指定したドライブ d ）中にあるディスク上へセーブを行います。

フロッピーディスクドライブが正しく接続されていなかったり、ディスクのライトプロテクトシールが付いている場合や、すでにあるファイル名を指定しようとした場合など、それぞれエラーが発生します。

プログラムテキストをカセットテープ上へセーブするには、**SAVE/T**コマンドを用います。

例　図2－2は、SAVEコマンドによって、スレーブディスク上に新しく1つのBASICテキストファイルが登録されるもようを示しています。左が、SAVEコマンドを行う前のディレクトリ表示、右がSAVEコマンドの実行とディレクトリ表示であり、BTXファイル"Statistic proc."が登録されたのがわかります。†）

```
DIR FD2
VOL.  17   1010 SECTOR FREE
 BTX   "3D-GRAPH"
 BTX   "19Angle RAMSAY"
 BTX   "17-Angle RAMSAY"
 BTX   "Sine curve"
 BTX   "Cycloid"
 BTX   "カンジ Editor"
Ready
```

図　2－2

†）ディレクトリ表示されるファイルの順番は、必ずしもファイルの登録順になるとは限りません。

2.1.4 LOAD

書　式　LOAD　<FD*d*@*v*, >　"*file name*"

d………ドライブナンバ：*d*＝1～4

v………スレーブディスク・ボリュームナンバ

機　能　BASICテキストエリアをクリアして、ディスク上に登録されているBASICテキストをロードします。

解　説　ロードすべきファイルは、FD *d* @ *v* によるドライブとボリュームナンバの指定、ファイル名の記述によって指定されます。デフォルトドライブを指定する場合はFD *d* @ *v* は省略できます。
ファイル名は、登録されているファイル名を省略なしで指定しなければなりません。
ディレクトリ表示を利用してカーソル操作を行うと、ファンクションキー10番を用いてワンタッチでファイルローディングを行うことができます。
たとえば、ディレクトリ表示上の次のファイル

BTX　　"Statistic proc."

をロードするには、カーソルを次のようにこの行の先頭へ持って行き、

▩BTX　　"Statistic proc."
↑
カーソル

ファンクションキー10番を押します。DEF KEY文によって新たに定義し直していなければ、ファンクションキー10番は、LOAD ┐が定義されているので、表示は、

LOAD　　"Statistic proc."

となり、"┐"記号のファンクション、即ちキャリッジリターンも実行されます。従ってワンタッチでLOADコマンドを与えたことになるわけです。
カセットテープ上のプログラムテキストをロードするには、**LOAD/T**コマンドを用います。

2.1.5 APPEND

書　式　APPEND　<FD*d*@*v*, >　"*file name*"

d………ドライブナンバ：*d*＝1～4

v………スレーブディスク・ボリュームナンバ

機　能　現在BASICテキストエリアにあるプログラムテキストと、指定したファイルのプログラムテキストを混ぜ合わせます。

解　説　指定したファイルの中に、BASICテキストエリアにあるプログラムと同じ行番号を持つものがあれば、テキストエリア上の行が対応するファイル上の行と置き換えられます。

例　APPEND FD2@47,　"TEXT-B"

………テキストエリアに次のプログラムが格納されており

```
10  A=345
30  S=10*A
40  PRINT S
```

ドライブナンバ2番中のスレーブディスク：ボリュームナンバ47番上にあるファイル名
"TEXT-B"の内容が次のとき

```
20  B=789
30  S=3*A-B
50  END
```

APPEND実行後のテキストエリア内のプログラムは次のようになります

```
10  A=345
20  B=789
30  S=3*A-B
40  PRINT S
50  END
```

2.1.6 RUN

書　式　RUN　<FD*d*@*v*, > "*file name*"

d………ドライブナンバ：*d*＝1～4

v………スレーブディスク・ボリュームナンバ

"*file name*"………BTXファイルまたはOBJファイル

機　能　BASICテキストエリアをクリアして、ディスク上に登録されているBASICテキストをロードし、続けてその先頭からプログラムを実行します。

OBJファイル（機械語オブジェクトファイル）へのRUNコマンドの実行は、DISK BASICの使用をやめ、該当OBJファイルにコントロールを移すことになります。

解　説　BTXファイルに対してRUNコマンドを与えると、次のマルチステートメント、

LOAD "*file name*"：RUN

を実行したのと同じく、ファイルのロードに続いて、そのプログラムの先頭から実行が行われます。

RUNコマンドは、ファンクションキー1番に定義されているので、LOADコマンドと同様にカーソル操作と共に使用すると便利です。

OBJファイルをRUNさせると、DISK BASICから、該当する別のシステムプログラムへコントロールが移ります。マスターディスク上の次の2つのファイルは、ユーティリティオブジェクトファイルであり、このRUNコマンドによって起動されます。

OBJ*　"Filing CMT"

OBJ*　"Utility"

2.2　ファイルコントロール文

2.2.1　LOCK

書　式　LOCK　〈FD*d*@ *v*,〉 "*file name*"

d………ドライブナンバ：*d* = 1 ～ 4

v………スレーブディスク・ボリュームナンバ

機　能　指定したファイルをロックします。

解　説　ファイルをロックすると、そのファイルはディスク上に固定された形になり、何らかの変更要求が来ても受けつけなくなります。

たとえば、DELETE、RENAME、ランダムアクセスファイル（BRD）の場合のデータ書き込みが禁止されます。壊したくないファイルや変更したくないファイルにはロックをかけておくようにします。

ロックされたファイルは、ディレクトリ表示で、ファイルモード記号につづいて"＊"マークが付きます。

たとえば、

LOCK　"Statistic proc."

コマンドの実行によってファイルがロックされ、ディレクトリ表示は、

BTX＊ "Statistic proc."
↑
ファイルロックを示す記号

となります。

ディスクのライトプロテクトシールは、ハード機構によってディスクへのデータ書き込みを禁止するものです。

2.2.2　UNLOCK

書　式　UNLOCK　〈FD*d*@*v*,〉 "*file name*"

d………ドライブナンバ：*d* = 1 ～ 4

v………スレーブディスク・ボリュームナンバ

機　能　指定したファイルのロックを解除します。

2.2.3 RENAME

書　式　RENAME　〈FD*d*@*v*,〉 "*file name*" , "*new file name* "

d………ドライブナンバ：*d*＝1～4

v………スレーブディスク・ボリュームナンバ

"*new file name*"………新規のファイル名

機　能　ファイル名の変更は、現在のファイル名、新しいファイル名の順に指定します。新しいファイル名は、そのディスク上にそれと同じファイル名で同一のモードのファイルが存在すればエラーとなります。

ロックされているファイルについてはRENAME命令は禁止されます。

2.2.4 DELETE

書　式　DELETE　〈FD*d*@*v*,〉 "*file name*"

d………ドライブナンバ：*d*＝1～4

v………スレーブディスク・ボリュームナンバ

機　能　指定したファイルをディスク上から削除します。

解　説　ロックされているファイルは、DELETEコマンドが禁止されます。ロックされているファイルを削除する場合は、UNLOCKコマンドを実行した後に、DELETEコマンドを実行します。

2.2.5 CHAIN

書　式　CHAIN <FD*d*@*v*,> "*file name*"

d………ドライブナンバ：*d*＝1～4

v………スレーブディスク・ボリュームナンバ

機　能　現在実行中のプログラムテキストから、ディスクファイル上の別のプログラムテキストへプログラム実行をチェインします。

解　説　CHAIN 文は、プログラム中でRUN"*file name*" コマンドを実行したのと似た働きがありますが、チェインする際、もとのプログラムで使用した変数、配列等の内容はそのまま新しいプログラムへ渡されます。従って、CHAIN 文は、GOTO *lr* 文と対照して GOTO"*file name*" のイメージとして捉えることができます。

例

100　CHAIN　FD2@7,　"TEXT　B"

…………ドライブ2番中のディスクがスレーブディスクのボリューム7番であれば、その中のBTXファイル"TEXT　B"へプログラム実行をチェインせよ、というステートメントです。

200　CHAIN　"Process　3"

…………デフォルトドライブ中のディスクに登録されている、BTXファイル"Process　3"へプログラム実行をチェインします。

2.2.6 SWAP

書　式　SWAP　<FD*d*@*v*,> "*file　name*"

d………ドライブナンバ：*d*＝1～4

v………スレーブディスク・ボリュームナンバ

機　能　プログラム実行をスワップします。

解　説　SWAP文を実行すると、最初に現在実行中のプログラムテキストをデフォルトドライブ（最後に DIR FD*d*コマンドを実行したドライブ）中のディスクに一旦待避（一時、記憶するだけで、ファイルを作るわけではありません）させ、オペランドに指定した BTX ファイルへコントロールが移されます。スワップされたプログラムが終了したら、もとのプログラムテキストをテキストエリアへ戻し、SWAP 文の次の実行文からプログラム実行を継続します。SWAP 文は、GOSUB"*file name*" のイメージで捉えることができます。ただ、スワップされたファイルからの復帰は、RETURN 文ではなく、END 文あるいは、テキストの終わりによります。

SWAP文はネスティングさせることができません(即ちレベルは1)。スワップされたテキスト内で更にSWAP文を実行しようとすると、*Error 25が発生します。

例

300　SWAP　FD3@27,　"TEXT　S-R1"

…………ドライブ3番中のディスクがボリュームナンバ27番のスレーブディスクであり、その上にBTXファイル"TEXT　S-R1"があれば、現在実行中のプログラムをデフォルトドライブ中へ待避し、"TEXT　S-R1"をローディングします。続いて、その先頭からプログラムを実行します。

END 文、プログラムテキストの終端によって実行を終了したら、もとのテキストを、デフォルトドライブから復帰させ、SWAP文の次の実行文から継続してプログラム実行を行います。

■BASICシーケンシャルアクセスデータファイル(BSD)コントロール文

2.2.7 WOPEN#

書式 WOPEN#*l*, 〈FD*d*@*v*,〉 "*file name*"

l………ロジカルナンバ：*l*＝1～127

d………ドライブナンバ：*d*＝1～4

v……… スレーブディスク・ボリュームナンバ

機能 1つのBASICシーケンシャルアクセスデータファイル（BSD）を作成するために書き込み用ファイルをオープンします。（WOPEN：write open）

解説 WOPEN#文は、シーケンシャルデータ登録のための準備を行う文であり、ファイルアクセス用ロジカルナンバの定義と、ファイル名の指定を行います。

例

```
510 WOPEN#3, FD2@10, "SEQ DATA1"
```

…………ドライブナンバ2番中のスレーブディスク：ボリュームナンバ10番上に、BSD"SEQ DATA1"を登録するため、ロジカルナンバ3番を書き込みオープンします。

2.2.8 PRINT#

書式 PRINT#*l*, d_1, 〈,d_2, …………, d_n〉

l………ロジカルナンバ

d_i………書き込みデータ並び

機能 WOPEN#文によって書き込みオープンされているファイル上へ、オペランドで指定するデータを順次書き込んで行きます。

解説 書き込みを行う対象となるBSDファイルは、ロジカルナンバ#*l*によって指定します。ディスクファイルは、前述（P.14）のように、10まで同時にオープンさせることができるので、目的とするファイルを、それをオープンした時のロジカルナンバで正しく指示しなくてはなりません。

例

```
510 WOPEN#3, FD2@10, "SEQ DATA 1"
530 FOR I=1 TO 30
540 PRINT #3, A(I), B(I), A$(I)
550 NEXT I
```

…………BSDファイル"SEQ DATA1"をロジカルナンバ3番に書き込みオープンし、Iを1から30まで変えながら順次、A(I)、B(I)、A$(I)の各配列要素の内容を書き込んで行きます。全部で90個のデータを書き込むことになります。

2.2.9 KILL

書　式　KILL　#l

l………ロジカルナンバ

機　能　ロジカルナンバl番にオープンしたBSDファイルの登録をキルします。即ち、WOPEN#文によってファイル作成を準備し、あるいはそれに続いてPRINT#を実行しているファイルの正式登録を途中でキャンセルします。

解　説　ロジカルナンバを指定しない場合は、現在作成中のファイルの正式登録をキャンセルするとともに、すでにオープンされているすべてのファイルをクローズし、そのロジカルナンバをすべて未定義番号に戻します。

なおKILL命令は直接実行命令として使用できますので、ディスクを交換する直前にこの命令を実行させ、すべてのファイルをクローズすることによってディスク内容の破壊より保護することができます。

2.2.10 CLOSE

書　式　CLOSE　#l

l………ロジカルナンバ

機　能　ロジカルナンバl番にオープンされているファイルをクローズして、この番号を未定義番号に戻します。

解　説　CLOSE文は、次のように、3種類のロジカルオープンに対して実行されます。

■ WOPEN#に対するCLOSE

WOPEN#、PRINT#文によるシーケンシャルアクセスデータの書き込みを終了して、このファイルを正式にディスク上へ登録します。使用したロジカルナンバを未定義に戻します。

■ ROPEN#に対するCLOSE

ROPEN#によってデータ読み出しオープンしたBSDファイルをクローズし、使用したロジカルナンバを未定義にします。

■ XOPEN#に対するCLOSE

XOPEN#によってデータ書き込み／読み出しオープンしたBRDファイルをクローズし、使用したロジカルナンバを未定義にします。

ロジカルナンバを指定しない場合は、現在オープンされているすべてのファイルをクローズして、そのロジカルナンバをすべて未定義番号に戻します。

例

```
510 WOPEN#3, FD2@10, "SEQ  DATA  1"
530 FOR  I=1  TO  30
540 PRINT  #3, A(I), B(I), A$(I)
550 NEXT  I
560 CLOSE  #3
```

…………行番号560のCLOSE#文によって、行番号510～550のループでデータを書き込んで行ったBSDファイル"SEQ　DATA　1"を正式にファイル登録し、クローズします。

2.2.11 ROPEN#

書　式　ROPEN# *l*, 〈FD*d*@*v*,〉 "*file name*"

l………ロジカルナンバ：*l* = 1 〜127

d………ドライブナンバ：*d* = 1 〜 4

v……… スレーブディスク・ボリュームナンバ

機　能　BASIC シーケンシャルアクセスデータファイル (BSD) 中のデータを読み出すためにファイルをオープンします。(ROPEN：read open)

解　説　ROPEN# 文は、シーケンシャルなデータ読み出しを行うための準備であり、読み出すべきファイルを指定し (〈FD*d*@*v*,〉"*file name*" によって指定します) それを、ロジカルナンバ *l* 番に設定します。

2.2.12 INPUT#

書　式　INPUT# *l*, v_1, 〈, v_2, ………, v_n〉

l………ロジカルナンバ

v_i………入力並び：変数または配列要素

機　能　ROPEN# 文によって、読み出しオープンされている BSD ファイルの先頭データから順次データを読み出し、オペランドの入力並びへ代入します。

解　説　データの読み出しを行うファイルは、ロジカルナンバ *l* 番にROPEN# 文を実行しているファイルになります。READ〜DATA文の場合と同様に、データと、入力並びのデータ型が一致しないとエラーが発生します。

例

```
700  ROPEN  #50, FD2@10, "SEQ  DATA  1"
710  FOR  I=1  TO  30
720  INPUT  #50, AA(I), BB(I), AA$(I)
730  PRINT/P "#No.";AA(I), " Vol." ; BB(I) , " Name : " ; AA$(I)
740  NEXT  I:CLOSE  #50
```

…………BSDファイル"SEQ DATA 1" (ドライブ2番中のボリューム10番のスレーブディスク内) に登録されているデータを行番号720のINPUT# 文で3個ずつ読み出し、その値をプリンタに整理して打ち出して行きます。

■BASICランダムアクセスデータファイル(BRD) コントロール文

2.2.13 XOPEN

書 式 XOPEN # *l*, <FD*d*@ *v*,> "*file name*"

l ………ロジカルナンバ：*l* = 1 ~127

d ………ドライブナンバ：*d* = 1 ~ 4

v ………スレーブディスク・ボリュームナンバ

機 能 1つの BASIC ランダムアクセスデータファイル (BRD) をオープンして、ランダムアクセスデータの書き込み／読み出しオープン (クロスオープン) を行います。(XOPEN：cross open)

解 説 XOPEN # 文は、1つの BRD ファイルの新規登録あるいは、既にある BRD からのデータ読み出し、或いは新しいデータの書き込みの準備を行い、ロジカルナンバ *l* 番を、ファイルアクセス用に定義します。

2.2.14 PRINT #( )

書 式 PRINT # *l*(*n*), d_1 <, d_2,………, d_n>

l ………ロジカルナンバ

n ………データ要素ナンバ

d_i ………書き込みデータ並び

機 能 XOPEN # 文によってクロスオープンされている BRD ファイルの要素 *n* 番から、書き込みデータ並びd_1～d_nの各データを書き込みます。

解 説 出力データ並びが1個の場合は、要素 (element) *n*番にそのデータが書き込まれますが、複数置かれた場合は、d_1を要素 *n* 番に、d_2を要素 *n* + 1 番へと順次書き込まれます。ランダムアクセスデータファイルの個々の要素は、すべて32バイト固定長であるので、ストリングデータの登録の場合、32キャラクタを越す分は、無効となるので注意を要します。

BRD ファイルにデータを書き込むには、データ要素ナンバを指定してそこへデータを書き入れますが、その要素ナンバを大きくする毎にその BRD のサイズは大きくなって行くことになります。

例

```
360  XOPEN  #73, "DATA  R73"
370  PRINT  #73 (22), R$ (22), CX
380  PRINT  #73(9), A(9), A(10), R$ (11)
390  CLOSE  #73
```

…………行番号370のPRINT # ( ) 文で、BRD ファイル "DATA R73" の要素22番にR$ (22) の内容が、要素23番に、変数CXの内容が書き込まれ、行番号380で、要素9番に、A(9)、要素10番にA(10)、そして要素11番にR$ (11) の内容がそれぞれ書き込まれることになります。

2.2.15 INPUT#( )

書式 INPUT# *l* (*n*), v_1 <, v_2, ………, v_n>

l ………ロジカルナンバ：*l* = 1 ~127

n ………データ要素ナンバ

v_i ………入力並び：変数または配列要素

機能 XOPEN # 文によってクロスオープンされている BRD ファイルの要素 *n* 番から登録されているデータを入力並びへ読み出します。

解説 入力並びが1個のときは、要素 *n* 番のデータが変数または配列要素v_1に読み出されますが、複数個ある場合は、要素 *n* 番のデータがv_1に、 *n* + 1 番のデータがv_2に、と順次読み出されます。

例

```
410 XOPEN #74, "DATA BRD4"
420 FOR J=1 TO 20
430 INPUT #74 (J*2), A(J)
440 NEXT J
450 CLOSE #74
```

…………行番号430で、BRD ファイル"DATA BRD4"の要素2 ~40の偶数番要素から数値データを読み出し、数値配列要素A(J) にそれぞれ代入しています。

2.2.16 IF EOF(#) THEN

書式 IF EOF(# *l*) THEN *lr* (または *statement*)

l ………ロジカルナンバ

lr………参照行番号

機能 BSD ファイルに対して INPUT # 文を実行したとき、あるいはBRD ファイルに対して INPUT #( ) 文を実行したときに、アウトオブファイルが発生した場合エラーは発生せず、入力並びには、0 または" " (null string) が入ります。IF EOF(#) THEN文はこの、アウトオブファイルが発生した場合の処理を定めるもので、各INPUT # 文の後に置きます。もしINPUT # 文でアウトオブファイルが生じていれば、THEN以下が実行されることになります。

例

```
800 ROPEN #3, "DATA"
810 INPUT #3, DT$ : PRINT DT$,
820 IF EOF(#3) THEN END
830 GOTO 810
```

…………BSD ファイル"DATA" に登録されているデータを最初から最後まで全部読み出して CRT ディスプレイ上へ表示させます。

2.3 エラー処理コントロール文

2.3.1 ON ERROR GOTO

書式 ON ERROR GOTO *lr*

lr………参照行番号（reference line number）：エラー処理ルーチンの先頭

機能 エラーが発生した時、エラー処理を行うためにプログラム実行を移す行番号を宣言します。

解説 ON ERROR GOTO 文によってエラー処理ルーチンを宣言することによって、エラー発生時に、BASIC コマンドレベルへ戻さずに、プログラム内でエラー処理を行うことが可能になります。

ON ERROR GOTO文が実行されていると、プログラム実行中にどのようなエラーが発生した場合もプログラム実行は *lr* で始まるエラー処理ルーチンへ移されて来ますが、このルーチンでIF ERN 文やIF ERL 文を用いることによってエラー番号（ERN）とエラー発生行番号（ERL）の判別ができ、それぞれ適当な処理を行うことができます。更に、エラー発生箇所にプログラム実行を戻すために、RESUME 文を使用することができます。

新たにON ERROR GOTO 文を実行すると、前のON ERROR GOTO 文は無効になります。

ON ERROR GOTOを宣言後、CLR命令を実行すると、この宣言は無効となります。

2.3.2 IF ERN

書式 IF ERN*expression* THEN *lr*

IF ERN*expression* THEN *statement*

IF ERN*expression* GOTO *lr*

ERN*expression*………ERNの関係式

lr………参照行番号

機能 発生したエラーの種類を判断して、エラー処理の分岐を行います。(ERN：error number)

解説 ON ERROR GOTO 文の実行によってエラー処理ルーチンの参照行番号が定義されている場合には、エラー発生時に、システム変数ERNに、エラー番号が代入されそのエラー処理ルーチンへコントロールが移されます。エラー番号は、巻末の表A－2に示されています。

IF ERN文は、エラー処理ルーチン内で、発生したエラーを判別するものであり、IF 文によってERNの値を調べることができます。IF ERN文は、書式に示されているように他のIF文と同様に3種類の形を使うことができます。

例 エラー処理ルーチンの先頭行番号を1000とし、そこで、エラー番号5（ストリングオーバーフローストリング長が255文字を越えた）であれば更に行番号1200へジャンプさせる例を次に示しています。

```
10 ON ERROR GOTO 1000………エラー処理ルーチンを定義する
…………………………
1000 IF ERN＝5  THEN 1200………ストリングオーバフロー・エラーであれば1200へジャンプする
…………………………
```

2.3.3 IF ERL

書式 IF ERL *expression* THEN *lr*

IF ERL *expression* THEN *statement*

IF ERL *expression* GOTO *lr*

ERL *expression*………ERLの関係式

lr………参照行番号

機能 発生したエラー箇所を判断して、エラー処理の分岐を行います。(ERL：error line number)

解説 エラー発生時に、システム変数ERLには、エラー発生行番号がセットされるので、ON ERROR GOTO文で宣言されたエラー処理ルーチンで、IF ERL文によるエラー発生箇所の判別がテキストの定義行番号によってできます。

IF ERL文は、IF ERN文と同じくIF～THEN、IF～GOTOのいずれの形でも使えます。

例 エラー発生行番号が250である時、1300行へ分岐するには、次の文を用います。

```
1010 IF ERL=250 THEN 1300
```

エラー番号が43でかつ、その発生箇所が450行上でなければ、メインプログラム中の行番号520へ戻すには次の文を用います。

```
1020 IF (ERN=43)*(ERL<>450) THEN RESUME 520
```

2.3.4 RESUME

書式 RESUME 〈NEXT〉

RESUME *lr*

lr………参照行番号または0

機能 エラー処理後、プログラム実行をメインプログラムへ戻します。

解説 エラー発生時にはエラー発生位置が記憶されているので、エラー処理を終了した後、その文あるいは、その次の文へプログラム実行を戻したり、或いは、他の任意の行へ戻すことができます。

即ち次の4通りの復帰文が記述できます。

RESUME………エラーが発生した文へ復帰します。

RESUME NEXT………エラーが発生した文の次の文へプログラム実行を戻します。

RESUME *lr*……… *lr*で指定する行へプログラム実行を戻します。

RESUME 0……… プログラムの先頭、即ち最小の行番号をもつ行へプログラム実行を戻します。

エラーが発生していないのにRESUME文に来た場合、ここでError 21 (RESUME-no ERROR)が発生します。

RESUME不能の場合、Error 20 (Can't RESUME) が発生します。

2.4 ユーティリティプログラムの使い方

DISK BASICマスターディスクには、BTXファイル"AUTO RUN"、幾つかのサンプルプログラム、それに、2つのユーティリティプログラムが用意されています。

2つのユーティリティプログラムは、次のようにいずれも機械語プログラムファイルです。(ディレクトリ表示)

OBJ* "Filing CMT"

OBJ* "Utility"

OBJファイル"Filing CMT"は、カセットテープファイル上のOBJファイルを、ディスク上へ移しかえるユーティリティプログラムであり、OBJファイル"Utility"は、ディスクのイニシャライズおよび、ディスクのコピー操作を行うためのユーティリティプログラムです。

2.4.1 ユーティリティプログラム"Filing CMT"の使い方

ユーティリティプログラム"Filing CMT"によってカセットテープファイルのOBJプログラム、たとえば、MONITORプログラムによって作成した機械語プログラムや、BASICインタープリタ MZ-1Z001などのシステムソフトウェアを、ディスク上へ登録することができます。ディスク上に登録されているファイルは、カセットファイルよりも早く読み出すことができます。

このユーティリティプログラムは、OBJファイルなので、DISK BASICからコントロールを移す必要があり、

RUN "Filing CMT"

コマンドを実行し、"Filing CMT"プログラムをスタートさせます。

"Filing CMT"プログラムがスタートすると図2－3に示す表示がなされます。

```
* TRANSFER FROM CMT (OBJECT TAPE) TO FD *
  SET TAPE! OK?
  (B KEY:BOOT START)
DRIVE NO.▩
```

図 2－3

ファイリングすべきカセットテープファイルをカセットデッキにセットし、転送先のドライブ番号をキーボードから入力します。これによってファイリングが自動的にスタートします。(カセットテープ上のOBJプログラムにはかならずファイル名が必要です。)

ファイリングが終了したら、次のメッセージを表示します。

```
(R) KEY:RESTART
OTHER KEY:BOOT START
```

即ち、続けて他のOBJファイルのファイリングをリスタートする場合 [R] キーを押し、そうでないときは他のキーを押してシステムのイニシャルプログラムローディングを行います。

2.4.2 ユーティリティプログラム"Utility"の使い方

RUN "Utility"

コマンドを実行することにより、ユーティリティプログラム"Utility"がスタートし、図2－4の表示をCRTディスプレイ上に行います。

```
** UTILITY **
〔COMMAND TABLE〕
DISK INIT          :I
SLAVE-DISK INIT    :S
DISK COPY          :C
BOOT START         :B
? ▒
```

図 2－4

即ち、I、S、Cの3つのユーティリティコマンドと、コントロールをIPLへ戻すBコマンドがこのプログラムにあります。

Iコマンド：DISK INITは、ディスクのイニシャライズを行います。すべての新しいディスクは、Iコマンドを実行して、ディスクのフォーマッティングを行なわなければなりません。

Iコマンドを実行すると、

```
? I
DRIVE NO. ▒
```

の表示がなされ、イニシャライズすべきディスクのセットしてあるドライブ番号を聞いてくるので指定して下さい。イニシャライズを終了すると、再び図2－4のコマンドテーブル表示がなされます。

Sコマンド：SLAVE-DISK INITは、Iコマンドによってイニシャライズしたディスクをスレーブディスクとして使用する場合に続けて実行し、スレーブディスク用のファイルテーブルと、スレーブディスク・ボリュームナンバの設定が行われます。

```
VOLUME NO.▒
```

のキー入力待ちに、1～127の範囲内で適当なボリュームナンバを指定します。ボリュームナンバは、ファイルコントロール文のオペランドに用いられ、スレーブディスクの判別に使われます。

Cコマンド：DISK COPYは、文字通り、1つのディスクの内容をイニシャライズした新しいディスクへコピーするプログラムです。これによって、次ページに述べるように、マスターディスクのサブマスターディスクを作成することができます。

Cコマンドを与えると、原本となるディスクのセットされているドライブ、続いてコピーを作成するディスクのセットされているドライブを聞いて来るので指定して下さい。たとえば、ドライブ1番のディスクの内容をドライブ2番へコピーするには、次のようにドライブ番号を与えます。

```
? C
FROM
DRIVE  NO.1
TO
DRIVE  NO.2
```

■サブマスターディスクの作成

Cコマンドを実行するとき、コピー原本をマスターディスクとしておくと、サブマスターディスクがコピーとして作られます。

ディスクは不注意な扱いによって使用不能となったり、不慮の事故、停電等によって壊されるということが起こり得ます。そういう場合を考えて、マスターディスクはいつも安全な場所に保管しておき、DISK BASICの使用は、普通このユーティリティによって作成したサブマスターディスクを使用するようにしてください。また万一のことを考えて、大切なスレーブディスクは必ずコピーを作っておくようにします。

"Utility"のCコマンドを用いて、サブマスターディスクをコピーすることはできません。図2－5にその関係を示しています。

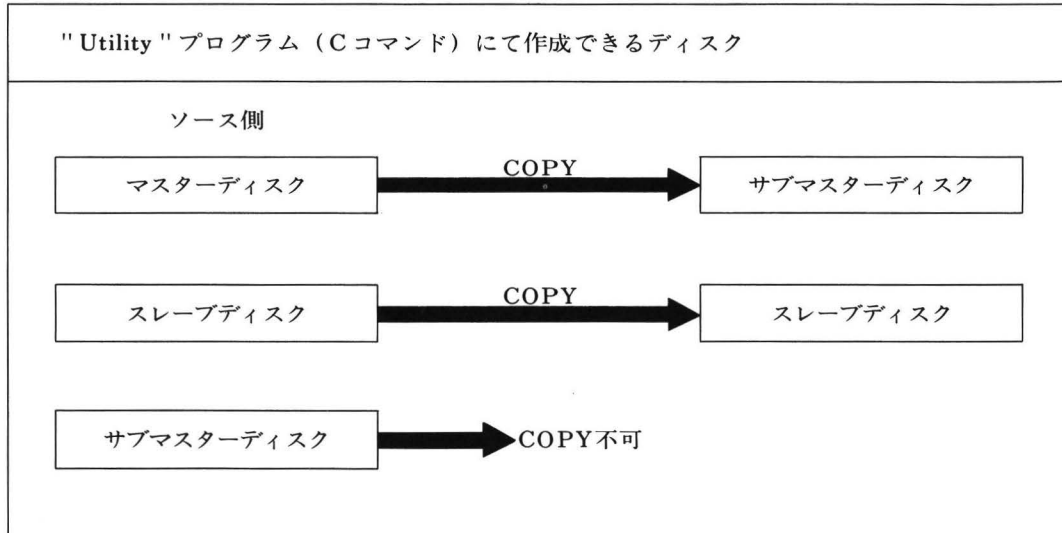



図　2－5

ユーティリティプログラムでのエラー表示

DISK　ERROR=50　………ディスクドライブがレディ状態にない

DISK　ERROR=41　………ディスクドライブのハード上のエラーが発生した

DISK BASIC応用プログラム データ処理アプリケーション

Chapter 3

この章は、DISK BASICを用いた典型的な応用プログラムを示しています。（なお、本プログラムを使用したことによる金銭上の損害および逸失利益または第三者からのいかなる請求についても当社はその責任を負いませんのであらかじめご了承ください。）
本プログラムはドットプリンタMZ-80BP5を対象として作成されています。

データ処理アプリケーション

このプログラムは、データファイルをユーザー自身で任意に決定することができ、そのファイルデータの演算処理、ソーティング、リスト作成等を可能とするものです。対象とするデータ処理、ファイルの種類に関し、多目的用途に応用できるように、汎用性をもっているため、成績表を作ったり、在庫管理にも用いることができます。このプログラムは次のジョブをもっています。

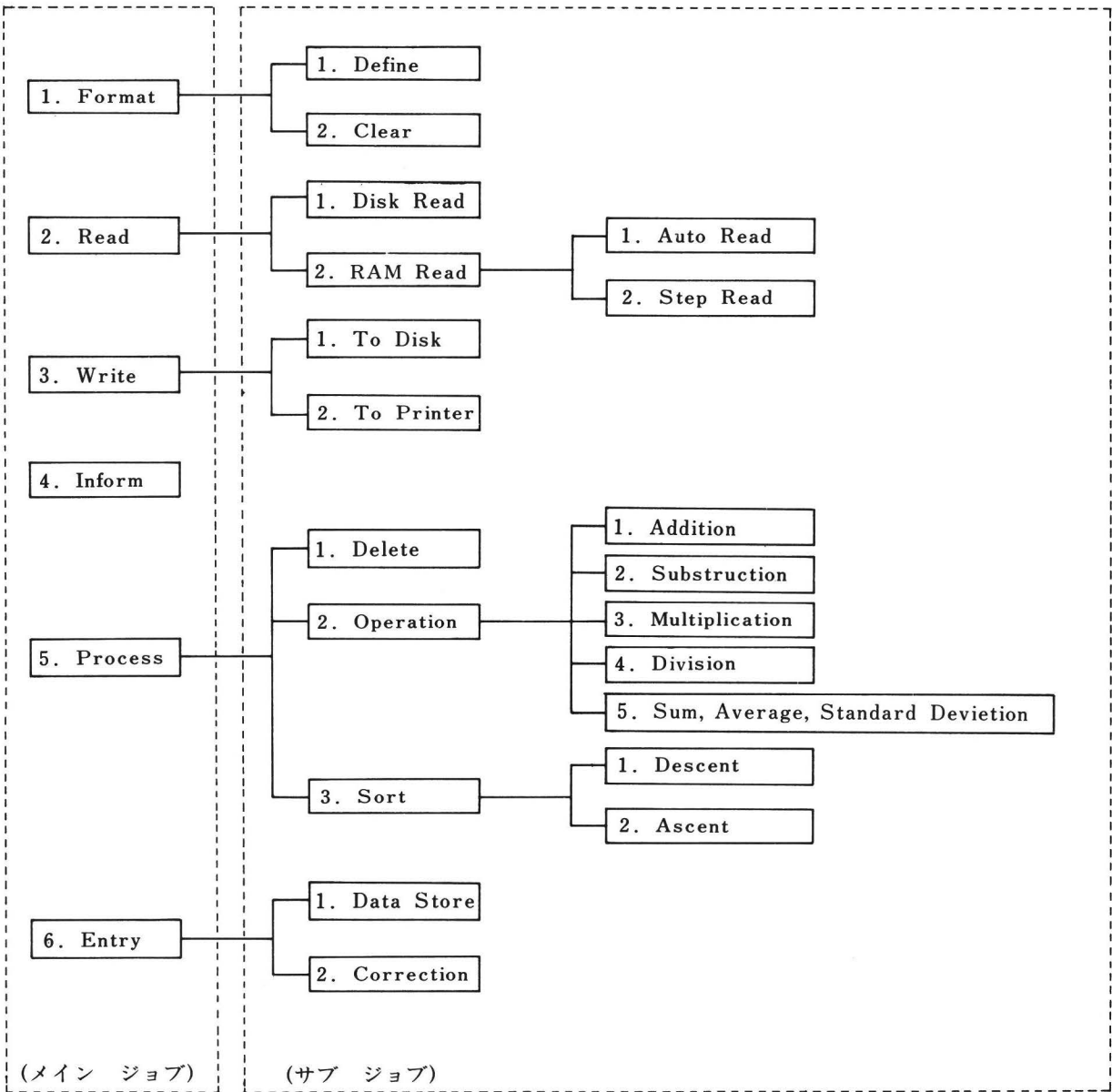



図　3—1

ファイルデータの構成

1つのファイルは下図のようにデータを構成します。フィールドのNは10以内であり、データの項目Mは200以内です。例えばデータ項目の2番目とは右図のアミのかかったデータ群を意味します。これはデータ入力時、このデータ群を1つとして登録することによります。従って、1項目の削除の場合は、やはりこの1まとまりのデータ群の削除を意味します。

項目 フィールド	1	2			N
1	データ	データ			データ
2	データ	データ			データ
3	データ	データ			データ
⋮	⋮	⋮			⋮
⋮	⋮	⋮			⋮
M-1					
M	データ	データ			データ

図 3−2

データの語長

1つのフィールドに対して入力データの語長を指定しなければなりません。これを決定するにはそのフィールドに属するデータのうちで最大の語長を、小なくともこのフィールドの語長として指定すべきです。

データの属性

入力データの属性とはそのデータが数値（Numeric）か、ストリングデータ（Alphanumeric）かということです。これを定義することにより、キー入力できる文字が制限されます。演算を行わせるフィールドのデータは必ずNumericとして定義しなければなりません。

Numeric モード　：　"＋"、"－"、"・"、0～9、スペース

Alphanumeric モード　：　アスキーコードの＄30～＄7Eの文字

ただし、両モード共に、カーソルコントロールキーの左右シフト、挿入、削除及び TAB キーのみを有効とします。

データの入力

カーソルがあらわれている場合のキー入力に対しては、データ入力後、必ず CR 又は ENT キーを入力しなければなりません。これによりデータは有効となります。このプログラムでは、すべてのキー入力時の任意の位置で TAB キーが有効であり、ジョブ選択ルーチン又はジョブ選択ルーチンへもどれる状態へ移行します。

ファイル格納ディスク

このプログラムによって作られたファイルを格納するディスクは複数であってもかまいません。ディスクからのファイルの読みだし、ディスクへのデータの書き込みの際、現在DIRにより選択されているディスクに対して動作します。即ち、途中でのディスクの交換が可能です。

また、このプログラムとファイルを格納するディスクは同一である必要はなく、このプログラム実効開始後の日付入力以降、ファイル格納用ディスクをディスクドライブに挿入します。

プログラムの実行

(a) 日付の設定

月、日、年の順に入力します。月と日に関しては、2文字まで、年に関しては、4文字までの数値とします。

例えば、1981年11月9日であれば

[1] [1] [ENT] → [9] [ENT] → [1] [9] [8] [1] [ENT]

または

[1] [1] [ENT] → [9] [ENT] → [8] [1] [ENT]

と入力します。図3－3 [ENT] のかわりに [CR] でもかまいません。

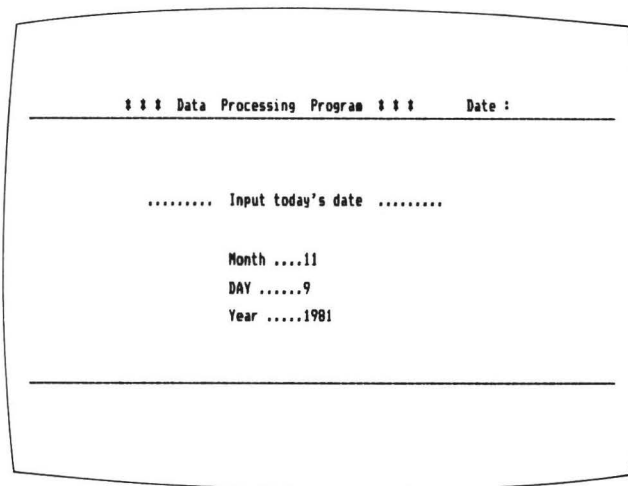



図 3－3

(b) 各ジョブの選択

CRTディスプレイ下部で"Main job"という文字がフラッシングしています。このとき、1～6のキーにより各ジョブを選択します。

メインジョブを選択した後、次にそのサブジョブを同じようにそれに対応するキーで選択します。図3－4

各ジョブについては次に説明を行います。

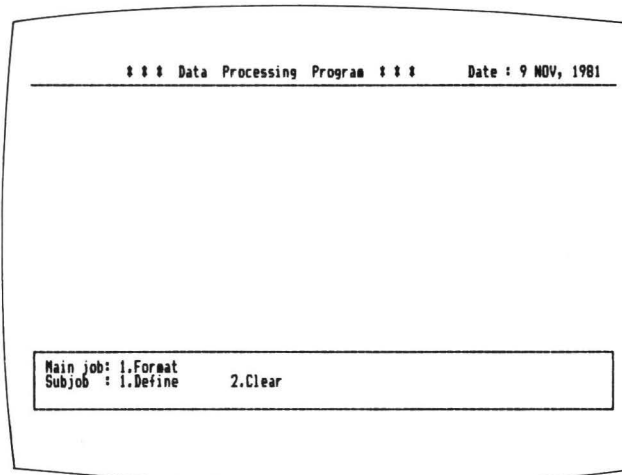



図 3－4

Job 1. Format

(1) Define

次の条件に従い、1フィールドに対するフィールド名、データの語長及びデータの属性を入力し、フィールドの定義を行います。データファイルを作成するとき、最初に行うべきジョブです。

フィールド名……………16文字以内の英数文字

データの語長……………79以内の数。ただし定義フィールドのデータ語長の総和は79以内

データの属性……………数値データ（Numeric）に対して"1"、ストリングデータ（Alphanumeric）に対しては"0"を入力

この操作を必要なフィールド数（10以内）だけ繰り返します。図3－5

図 3—5

(2) Clear

定義済みフィールド及びそのフィールドのデータをすべて消去し、初期状態に戻します。新しいファイルを作成するときなどに用い、未定義フィールドとします。

Job 2. Read

(1) Disk Read

ディスクに格納されているデータファイルを読み出します。CRTディスプレイに格納ファイルの情報（ファイル名、格納日付等）を表示するので、読み出すファイルがあれば、それに対応する番号で指定します。

1画面に表示するファイル数は9以内ですが、"To be continued" の表示があれば、ディスクにはさらに格納ファイルが存在することを意味します。このとき、スペースキーを押すことにより、次の格納ファイルの情報を表示します。"End" の表示があれば次に表示すべきファイルは存在しないことを意味します。図3－6

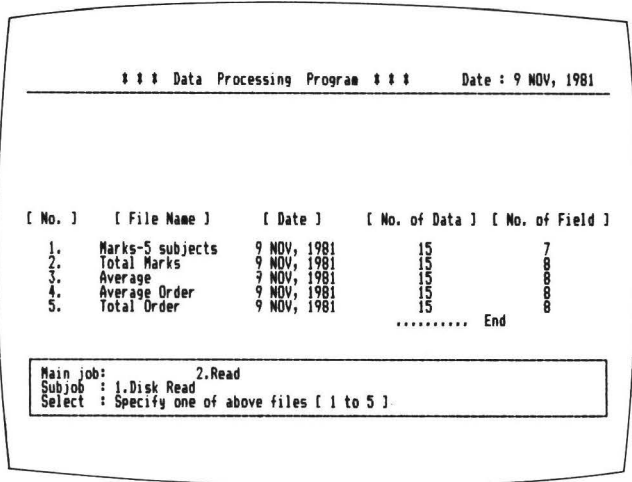



図　3－6

(2) RAM Read

内部メモリにあるデータファイルの各データをCRTディスプレイに表示します。連続表示を可能とする"Auto Read"と1ステップごとにデータを表示します"Step Read"の2つのジョブがあります。

"Auto Read"中、スペースキーが有効であり、表時の1時中止、又は1時中止しているリスティングを再開します。図3－7

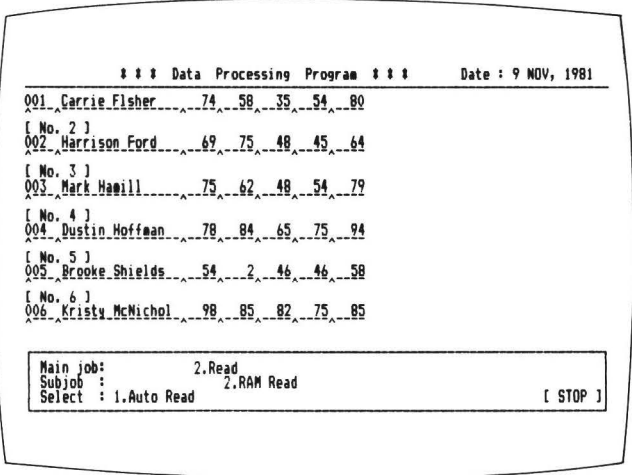



図　3－7

"Step Read"中は、スペースキーを入力するごとに次のデータを表示し、次のスペースキーの入力を待ちます。図3－8

両モードにおいて任意の時にスペースキー、"1"、"2"のキーが有効です。

図 3－8

Job 3. Write

(1) To Disk

16文字以内でファイル名を指定し、データをディスクに格納します。すでにディスクに格納されているものと同一のファイル名を指定した場合、ディスクのそのファイルを削除すべきか否かを問い正しくるので、"1"又は"2"のキー入力で応答します。"1"の入力はディスク上のそのファイルを削除し、新たに、内部メモリにあるファイルをディスクに書き込みます。

"2"の入力はファイル名の再指定を行う場合です。図3－9

図 3－9

(2) To Printer

内部メモリ上のデータファイルをプリンタに出力します。ファイル名を16文字以内で最初に入力します。リスティングを途中で中止したい場合は TAB キーを押すことによりジョブ選択ルーチンに戻ることができます。

Job 4. Inform

ジョブ1の"Define"で定義したフィールドの定義状態をCRTディスプレイに表示します。定義状態の確認後、任意のキー入力によりジョブ選択ルーチンに戻ります。図3－10

図 3－10

Job 5. Process

(1) Delete

内部メモリ上のファイルから、不用のデータを項目単位で削除します。まず、削除する項目の先頭番地を指定し、次に、その項目を含め、何項目分削除するかを決め、その個数を入力します。

削除後は、削除項目以降にある項目は前につめられます。図3－11

図 3－11

(2) Operation

1～4のキーは、それぞれフィールド・フィールド間の加算、減算、乗算、除算のジョブに対応します。図3－12

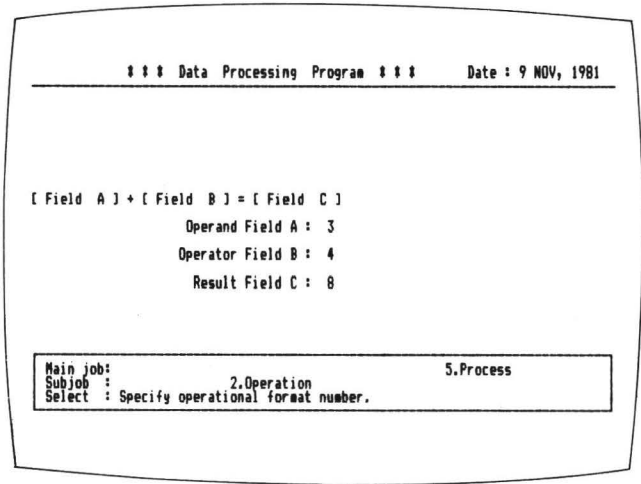



図 3－12

例えば、加算を行うときには、まず"1"のキーで加算ジョブを選びます。次に、演算フィールドと被演算フィールド及びその結果を格納するフィールドを入力します。この際の入力はフィールド番号とします。図3－13はフィールド3とフィールド4のデータの和をフィールド8に格納させることを意味します。

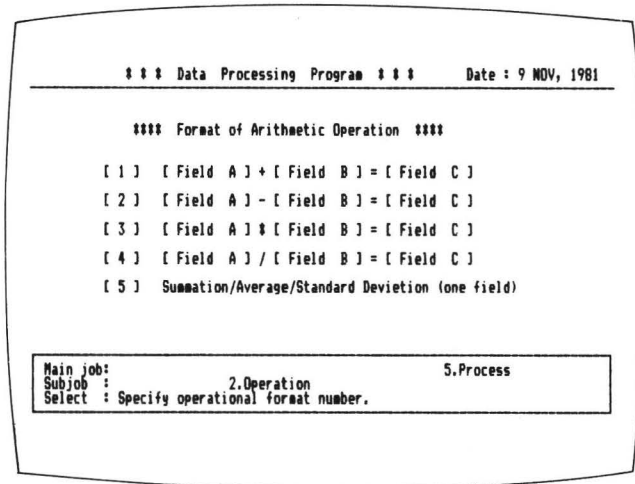



図 3－13

演算フィールド又は被演算フィールドの指定の際、負の数の入力（例えば－1）を入力すると、そのフィールドのデータを一時的にある定数値とすることができます。負の数を入力後、"Constant Value"と表示するので、任意の定数値を入力します。

図３－14は、フィールド８のデータを定数５で割り、その結果をフィールド８に格納させることを意味します。

```
*** Data Processing Program ***        Date : 9 NOV, 1981

[ Field A ] / [ Field B ] = [ Field C ]
              Operand Field A : 8
              Constant Value  : 5
               Result Field C : 8

Main job:                                  5.Process
Subjob  :            2.Operation
Select  : Specify operational format number.
```

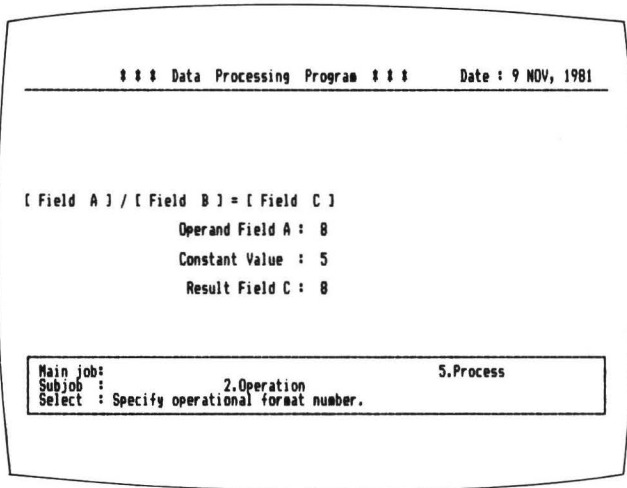



図　3―14

ここで注意すべきこととして、演算結果を格納するフィールドのデータ語長は、その結果に見あった語長として定義しなければなりません。演算結果はすべて小数第１位までとし、それ以降は無視します。もし演算結果の語長が、格納フィールドの語長を越えた場合、このデータは無効とします。また、フィールドの指定は、必ずNumericフィールドでなくてはなりません。

"５"のキー入力は、１つのフィールドに対する総和、平均値、標準偏差を求めるジョブの指定です。ここでのフィールド指定はNumeric フィールドとし、演算結果は小数第２位までとします。図３－15

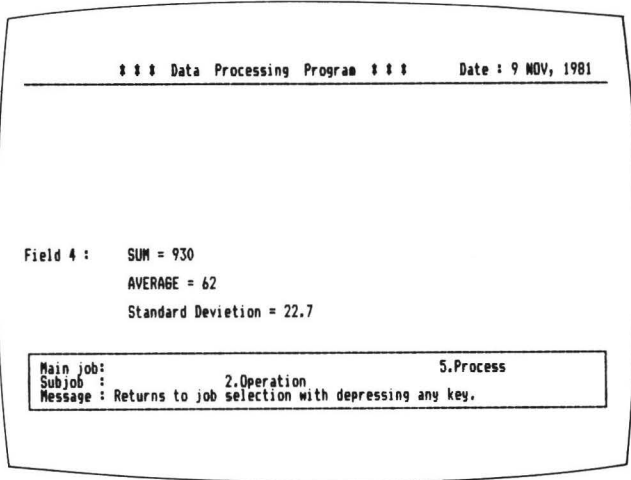



図　3―15

(3) Sort

指定のNumericフィールドに関し、ソートを行います。値の大きい順に並べかえる"Descent"と値の小さい順に並べかえる"Ascent"の２種類があります。それぞれ１、２のキーで指定します。この指定後、ソートフィールドを番号で指定します。図３－16

図　3－16

Job 6. Data input

(1) Data Store

定義フィールドの内容によって、それぞれのフィールドのデータを入力します。CRTディスプレイに例えば次の表示がされます。

"＾"はそのフィールドのデータの書き始めを意味し、次の"＾"の前までが、そのフィールドの語長となります。データは指定された場所を越えて入力はできません。また、Alphanumericフィールドはスペースも１つの文字として扱い、Numericではスペースは無視します。１つのフィールドのデータを入力し終えたなら、CR 又は ENT キーを押し、次のフィールドの入力に入ります。データ入力の際、そのフィールドの定義状態がCRTディスプレイ下部に表示されます。図３－17

図 3−17

(2) Correction

入力データの訂正を行うジョブであり、訂正項目番号を指定することにより、その項目の各フィールドのデータがCRTディスプレイに表示されます。このときカーソルはフィールド１に位置しているので、フィールド５のデータを訂正するには CR 又は ENT キーを４回押しフィールド５の位置にカーソルを移動させ、データの訂正を行います。訂正後は必ず次のフィールドにカーソルを移動させ、訂正データを有効にさせます。訂正が終れば、 TAB キーを入力します。図３−18

* * * Data Processing Program * * *　Date : 9 NOV, 1981

Specify correction item number (Within 17) :17

No. 17
017 James Caan 54 72 66 69 72

Main job:　6.Entory
Subjob :　2.Correction
Message : Field 5: Mathematics　Numeric　Length= 5

図 3−18

プログラム中の主な変数

AA	：ディスクに格納されるファイルの数
DT＄	：日付
NF	：定義フィールドの数　(N1＝NF－1)
P1	：登録された項目数　(P1＝P－1)
A＄(J、I)	：A＄(J、0)は(J＋1)番目の定義フィールドのフィールド名を格納 I>1において、A＄(J、I)は(J＋1)番目のI項目のデータを格納
B(I)	：(I＋1)番目のフィールドのデータ語長
A(I)	：(I＋1)番目のフィールドの属性
U	：登録フィールドのデータ語長の和

機械語ルーチン

メモリアドレス＄FE6C～＄FFFFが機械語領域であり、3つのプログラムから構成されます。

(1)　ROPEN＃1、USR(＄FF7A)

BASIC領域からストリング変数IP＄にデータをもって、アドレス＄FF7Aをコールし、キーボードからの入力データをIP＄に代入してリターンします。このルーチンをコールする前に次のデータをセットします。

アドレス＄FFFE、＄FFFF	：V-RAMアドレス、ここでは＄D550
アドレス＄FFFC	：入力データの書き始めを上記V-RAMアドレスからの変位で与える
アドレス＄FFFB	：入力データの語長
アドレス＄FFFA	：入力データの属性（1＝Numeric、0＝Alphanumeric）

(2)　USR(＄FFDD)

1行80文字のモードでのCRTディスプレイに対し、3行目から20行目（トップラインを1行目とします。）に表示されたデータを1行分上にシフトした状態の表示を行います。ただし20行目はスペースラインします。

(3)　USR(＄FFEA)

押したキーに対応するアスキーコードを1文字分だけ、アドレス＄FFF9に書き込んでリターンします。

ディスクファイルの構成

作成されたディスクファイルは任意のプログラム名をもってディスクに書き込まれます。1つのファイルはシリアルデータとして構成されています。また、各ファイルの格納情報として、ファイル名、登録日付、定義フィールド数、格納項目数をランダムファイル"Storage File"に格納しています。

フローチャート

メイン
プログラム
スタート
CRTへの表示を
80characters
/line
のモードに設定
配列変数の宣言
エラー発生時の
処理ルーチンの
宣言
ファイル名
" Constant Data "
ストリング変数
機械語データを
ディスクから読
み出す。
日付入力
有効か？
メインジョブ
選択
有効か？
" Format "
のルーチンか？
" Format "
ルーチンへ
GOTO 3000
サブジョブの
選択
GOTO 1540
" Define " ルーチンへ
GOTO 1530
" Clear " ルーチンへ
" Format "
GOTO 2000
" Disk Read " ルーチンへ
GOTO 2250
" RAM Read " ルーチンへ
" Read "
GOTO 2500
" To Disk " ルーチンへ
GOTO 2650
" To Printer " ルーチンへ
" Write "
GOTO 4000
" Delete " ルーチンへ
GOTO 4200
" Sort " ルーチンへ
GOTO 4500
" Operation " ルーチンへ
" Process "
GOTO 3500
" Data Store " ルーチンへ
GOTO 3600
" Correction " ルーチンへ
" Entry "

"Define"
スタート
(1540～1740)
定義可能なフィールドはあるか？
No
メッセージ
GOTO 1000
FOR I=NF TO 9
NF－1：定義済フィールド数
FOR J=0 TO 2
J=0：フィールド名
J=1：データ語長
J=2：データ属性
語長の和≧79 ？
Yes
入力条件表示
メッセージ
キー入力条件設定
キー入力ルーチン
TAB キー？
GOTO 1000
No
J=1
J=0
J=2
No
No
有効データか？
フィールド名登録
有効データか？
Yes
語長登録
属性登録
NEXT J
登録フィールド数をインクリメント
NEXT I
追加フィールドであったか？
No
ダミーデータの書き込み
GOTO 1000
"Clear"
スタート
フィールド変数の初期化
GOTO 1000

"Disk Read"
スタート
(2000～2200)
ディスクにデータファイルが存在するか？
メッセージ
GOTO 1000
ディスクから格納ファイルのインフォメーションデータ読み出し
"Storage File"
ファイル名、日付、データ数、定義フィールド数表示
他にファイルがあるか？
No
9つのファイルのインフォメーションを表示したか？
No
まだファイルが存在するか？
No
"To be continued"
"End"
ファイル読み出しのキー入力条件設定
キー入力ルーチン
TAB キー？
Yes
スペースキー？
Yes
インフォメーションファイルのデータはまだ存在するか？
指定ファイルをディスクから読み出す
GOTO 1000

"RAM Read"
スタート
(2250～2375)
ファイルにデータは存在するか?
No
メッセージ
GOTO 1000
"Auto Read"か
"Step Read"
かの選択
(キー入力)
TAB キー?
Yes
GOTO 1000
指定の読み出し
モードに設定
全項目表示したか?
Yes
最初の項目にもどす
項目データ表示
モード切り換え
キー入力
キー入力はあったか?
No
"Auto Read"モードか?
TAB キー?
Yes
GOTO 1000
スペースキー?
Yes
"Auto Read"モード指定か?
Yes
"Auto Read"
モードに設定
"Step Read"
モードに設定
表示を一時
中断

"To Disk"
スタート
(2500～2625)
ファイルデータは存在するか？
No
メッセージ
Yes
ファイル名の指定
GOTO 1000
Yes
TAB キー？
No
同一ファイル名はあるか？
Yes
エラーメッセッジ
No
ファイルのインフォメーションをディスクに書き込む
ディスクのファイルを削除？
No
Yes
ファイル削除
指定のファイル名をもってディスクにデータファイル書き込む
GOTO 1000
"To Printer"
スタート
(2650～2840)
ファイル名入力
TAB キー？
Yes
No
定義フィールドのフィールド名を印字
項目データ印字
TAB キー？
Yes
No
全項目印字したか？
GOTO 1000

"Inform"
スタート
(3000〜3090)
定義された
フィールド
は存在するか？
No
メッセージ
GOTO 1000
定義フィールド
のフィールド名、
語長、属性を表
示
定義された
フィールド
をすべて表
示したか？
No
残りのフィール
ドの定義状態を
スペース表示
キー入力
キーが
押されたか？
GOTO 1000

"Delete"
スタート
(4000～4120)
ファイルデータは存在するか？
No
メッセージ
削除項目の先頭番号指定
TAB キー？
Yes
GOTO 1000
No
有効か？
削除項目数入力
TAB キー？
Yes
削除項目数の調整
No
削除項目以降にデータがあるか？
削除項目の部分へデータをシフトさせる

" Operation "
スタート
(4500～5050)
演算フォーマットの表示
演算フォーマットの選択
Yes
GOTO 1000
TAB キー?
No
フィールド間の演算か?
No
Yes
演算フィールドの指定
TAB キー?
Yes
総和、平均標準偏差の計算
結果表示
キー入力はあったか?
No
Yes
GOTO 1000
被演算フィールド、演算フィールド、格納フィールドの指定
Yes
GOTO 1000
TAB キー?
No
負の数?
Yes
定数値入力
No
有効フィールドか?
No
指定の演算実行
演算結果の語長は有効か?
無効データを設定
演算結果を格納
No
すべての項目の演算を終えたか?
GOTO 1000

"Sort"
スタート
(4200～4490)
"Descend"
又は
"Ascend"
の選択
TAB キー？
Yes
指定のモード
に設定
ソートするフィ
ールドの指定
TAB キー？
Yes
GOTO 1000
No
有効か？
ソートフィール
ドの初期スタッ
ク設定
スタック
プッシュ
スタック
ポップ
データの変換区
間の設定
交換の基準値
の設定
データの交換
処理
交換区間
の細分
スタック
プッシュ
No
初期スタック
状態にもどっ
たか？
GOTO 1000

" Entry "
スタート
(3500～3560)
定義された
フィールド
はあるか？
No
メッセージ
登録データ
数はMaxか？
Yes
GOTO 1000
データの項目
番号表示
メッセージ
FOR I=0
TO N1
GOTO 1000
キー入力
ルーチン
TAB キー？
Yes
GOTO 1000
NEXT I
登録項目
数を決定
登録
項目数は
Max？
Yes
メーセージ
GOTO 1000

" Correction "
スタート
(3600～3700)
訂正する項目
番号のキー入力
TAB キー？
GOTO 1000
No
有効か？
指定項目の
データ表示
FOR I=0
TO N1
データの
キー入力
TAB キー？
NEXT I

```
1 REM........................ Constant Data TX ..............................
2 REM
3 CONSOLEC80
4 PRINT".......        This program makes the serial data file 'Constant Data'
5 PRINT".......        which is used in the main program 'Data  Processing'.
6 PRINT".......        Before executing the main program,   the above file
7 PRINT".......        'Constant Data' must be stored in a disk in
8 PRINT".......        which the main program is stored.
9 REM
10 DIM F$(2),K$(1),LM(2),FF$(4),FM$(6),FS$(4),ER$(14),S$(2),O$(4)
40 MD$="JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDEC":R$="Storage File"
41 O$=" Field  ":O$(0)="["+O$+"A ] + ["+O$+"B ] = ["+O$+"C ]"
42 O$(1)="["+O$+"A ] - ["+O$+"B ] = ["+O$+"C ]"
43 O$(3)="["+O$+"A ] / ["+O$+"B ] = ["+O$+"C ]":O$(4)="Summation/Average/Standard Deviation (one field)"
44 S$(0)=SPACE$(79):S$(1)=STRING$(".",79):S$(2)="|"+SPACE$(77)+"|"
45 LM(0)=16:LM(1)=2:LM(2)=1:O$(2)="["+O$+"A ] # ["+O$+"B ] = ["+O$+"C ]"
50 GT$="16121212001312":K$(0)="Alphanumeric":K$(1)="Numeric"
70 DK$="Month ....DAY ......Year ....."
110 F$(0)="Field Name     (Up to 16 characters)"
120 F$(1)="Field Length   (Total Length < 80)"
130 F$(2)="Field Attribute (0=Alphanumeric, 1=Numeric)"
150 FR$="|              "
160 FF$(0)="| Main job: ":FF$(1)="| Subjob  : ":FF$(2)="| Select  : "
170 FF$(3)="| Error   : ":FF$(4)="| Message : "
180 FM$(0)="1.Format   2.Read     3.Write    4.Inform   5.Process  6.Entry   |"
190 FM$(1)="1.Define        2.Clear      "
200 FM$(2)="1.Disk Read     2.RAM Read   "
210 FM$(3)="1.To Disk       2.To Printer "
220 FM$(5)="1.Delete        2.Operation    3.Sort         "
230 FM$(6)="1.Data Store    2.Correction "
240 FS$(0)="Specify one of above files [ 1 to   ]"
250 FS$(1)="1.Auto Read     2.Step Read  "
260 FS$(2)="1.Descent       2.Ascent     "
270 FS$(3)="Specify operational format number."
280 FS$(4)="Returns to job selection with depressing any key."
290 ER$(0)="Invalid ....... No Data  or  No File"
300 ER$(1)="Invalid Data"
310 ER$(2)="Total length is over 79.":ER$(5)="File is filled."
320 ER$(3)="Total length is fulled.":ER$(6)="Impossible ... Alphanumeric Field"
325 ER$(4)="Fields are filled."
330 ER$(7)="No file to be stored"
340 ER$(8)="Printer is not ready.":ER$(9)="Printer is in mechanical trouble."
350 ER$(10)="Printer is out of paper.":ER$(11)="Disk drive is not ready."
360 ER$(12)="No memory capacity to be stored on the disk":ER$(13)="Disk is not initialized."
370 ER$(14)="Same file name exists."
380 WOPEN#1,"Constant Data":PRINT#1,GT$,MD$,R$,DK$,FR$,K$(0),K$(1)
390 FORI=0TO2:PRINT#1,S$(I),LM(I),F$(I):NEXT
400 FOR I=0 TO 4:PRINT#1,FF$(I),FS$(I),O$(I):NEXT
410 FOR I=0 TO 6:PRINT#1,FM$(I):NEXT
440 FOR I=0 TO 14:PRINT#1,ER$(I):NEXT
500 FORI=0TO395:READ A:PRINT#1,A:NEXT:CLOSE:END
700 DATA 62,31,18,1,1,0,201,221,126,0,195,213,254,221,33,108,254,213
710 DATA 42,254,255,58,252,255,95,22,0,25,34,252,255,58,251,255,95,229
720 DATA 25,43,34,254,255,58,249,255,183,40,11,62,32,67,35,4,205,80
730 DATA 12,43,16,250,225,1,255,32,58,250,255,183,40,3,1,58,48,62
740 DATA 250,8,123,186,32,31,205,50,8,205,176,255,205,214,255,8,61,32
750 DATA 238,205,62,12,254,31,202,116,254,221,119,0,62,31,205,80,12,24
760 DATA 218,205,138,255,24,226,205,80,12,35,20,201,122,183,202,131,255,21
770 DATA 205,116,255,43,201,123,186,202,131,255,205,116,255,35,20,201,122,183
```

```
780 DATA 202,131,255,123,146,254,0,40,228,197,79,6,0,229,213,205,116,255
790 DATA 84,93,27,205,93,12,43,62,32,205,80,12,209,225,43,21,193,201
800 DATA 123,186,40,94,229,42,254,255,205,62,12,254,32,225,32,82,205,116
810 DATA 255,123,61,146,254,0,40,186,197,71,42,254,255,43,205,62,12,35
820 DATA 205,80,12,43,16,245,62,32,205,80,12,193,201,209,209,205,116,255
830 DATA 195,109,254,0,193,205,116,255,42,252,255,75,6,0,209,213,197,205
840 DATA 62,12,18,35,19,13,32,247,193,209,201,205,62,12,254,31,192,221
850 DATA 126,0,205,80,12,201,0,0,205,20,15,205,20,15,201,205,50,8
860 DATA 184,56,4,185,218,223,254,245,58,250,255,183,40,56,241,254,43,202
870 DATA 223,254,254,45,202,223,254,254,46,202,223,254,254,32,202,223,254,254
880 DATA 0,200,254,13,40,164,254,27,40,151,254,4,202,229,254,254,7,202
890 DATA 251,254,254,3,202,240,254,254,8,202,33,255,205,131,255,201,241,24
900 DATA 218,197,6,96,16,254,193,201,1,80,5,17,160,208,33,240,208,205
910 DATA 93,12,201,6,160,205,50,8,254,0,32,2,16,247,50,249,255,201
```

```
1 REM............     DATA  PROCESSING  MAIN  PROGRAM      .................
2 REM
3 CONSOLEC80
4 CURSOR0,15:PRINT"(Note)  There are 3 files -- 'Data Processing', 'Constant Data TX'"
5 PRINT"         and 'Constant Data' -- on the Master Disk."
6 PRINT"         But the files 'Data Processing' and 'Constant Data' should"
7 PRINT"         be transferred from the Master Disk to a slave disk."
8 PRINT"         Otherwise, error will cause under the execution."
9 PRINT"         With executing the program 'Constant Data TX',  the serial"
10 PRINT"         data file 'Constant Data' will be made."
20 LIMIT$FE6C:KILL:DIMA$(9,200),A(9),B(9),F$(2),K$(1),LM(2),FF$(4),FM$(6),FS$(4),ER$(14)
30 DIMS$(2),E(2),O$(4),X(1,100):A3=1:TEMPO6:P=1:P1=0:ON ERROR GOTO6000
100 ROPEN#1,"Constant Data":INPUT#1,GT$,MD$,R$,DK$,FR$,K$(0),K$(1):FORI=0TO2:INPUT#1,S$(I),LM(I),F$(I):NEXT
120 FORI=0TO4:INPUT#1,FF$(I),FS$(I),O$(I):NEXT:FORI=0TO6:INPUT#1,FM$(I):NEXT
130 FORI=0TO14:INPUT#1,ER$(I):NEXT
200 X=65133:FORI=0TO395:INPUT#1,A:POKEX+I,A:NEXT:CLOSE
500 PRINTCHR$(6);TAB(13);"* * *  Data  Processing  Program  * * *";TAB(59);"Date : ":PRINTSTRING$("-",79)
510 CURSOR0,20:PRINTSTRING$("-",79)
700 CURSOR16,17:PRINT".........  Input today's date  .........":A7=3:GOSUB7080
710 FORI=0TO2
720 CURSOR27,17:PRINTMID$(DK$,10*I+1,10):A4=1:A5=2:IFI=2THENA5=4
730 A6=37:GOSUB7600:IFD1=0GOSUB7070:GOTO700
740 IFIP<1THEN7930
750 ONIGOTO800,820
760 IFIP>12GOTO7930
770 X=IP:GOTO820
800 Y=IP:IFY>31GOTO7930
805 IF(X=2)*(Y>29)GOTO7930
810 IF((X=4)+(X=6)+(X=9)+(X=11))*(Y=31)GOTO7930
820 A7=1:GOSUB7080:NEXT:DT$=LEFT$(STR$(Y)+" "+MID$(MD$,3*X-2,3)+", "+STR$(IP)+S$(0),12):CURSOR0,0:PRINTTAB(78-LEN
(DT$));DT$
1000 GOSUB7070:CURSOR0,20:PRINT"┌";STRING$("-",77);"┐":PRINTFF$(0);FM$(0):PRINTS$(2):PRINTS$(2):PRINT"└";STRING$("-",
77);"┘";
1010 CP=21:GF=0:K=0:JM=0:GOSUB7100:JM=G:IFG3=0GOTO1010
1130 CURSOR12,21:PRINTSPACE$(11*(G-1));TAB(12+11*G);SPACE$(11*(6-G)):IFJM=4GOTO3000
1160 CURSOR0,22:PRINTFF$(1);FM$(JM):CP=22:GF=0:K=1:GOSUB7100:SJ=G:IFG3=0GOTO1000
1200 CURSOR12,22:PRINTSPACE$(15*(G-1));TAB(12+15*G);SPACE$(11*(3-G))
1210 ONJMGOTO1500,2000,2500,3000,4000,3500
1250 ER=0
1260 GOSUB7400:GOTO1000
1500 IFSJ=1GOTO1540
1530 GOSUB7920:PRINT"clearing buffer !!":FORL1=0TON1:FORL2=0TOP1:A$(L1,L2)="":NEXTL2,L1:P=1:NF=0:P1=0:N1=0:U=0:GOTO1000
1540 IF(NF>=10)+(U>=79)THENER=5:GOTO1260
1545 Y=NF:A6=37:ZZ=0
1560 FORI=NFTO9:FORJ=0TO2:IF(J=0)*(U=79)GOTO1695
1580 CP=23:GOSUB7700:PRINTFF$(4);"Input ";F$(J);:A4=SGN(J):A5=LM(J)
1585 IFJ=1THENPRINTTAB(62);"[Remainder:";79-U;"]"
1590 IFJ=0THENCURSOR0,17:PRINT"[ Field :";I+1;" ]"
1600 CURSOR15,17:PRINTLEFT$(F$(J),15);" =":GOSUB7600:IFD1=0GOTO1700
1610 ONJGOTO1630,1660
1620 A$(I,0)=IP$:GOTO1680
1630 IFIP<=0THENER=1:GOSUB7400:GOTO1590
1640 U=U+IP:IFU>79THENU=U-IP:ER=2:GOSUB7400:GOTO1590
1650 B(I)=IP:GOTO1680
1660 A(I)=IP:IF(IP<>0)*(IP<>1)THENER=1:GOSUB7400:GOTO1580
1680 A7=2:GOSUB7080:NEXTJ
1690 A7=3:GOSUB7080:NF=NF+1:N1=NF-1:NEXTI
1695 CP=23:GOSUB7700:PRINTFF$(4);ER$(4):K=4:CP=23:GF=2:GOSUB7110
1700 IFY=NFGOTO1000
```

```
1705 IFP=1GOTO1000
1710 FORI=1TOP1:FORJ=YTON1:IFA(J)=0THENA$(J,I)=SPACE$(B(J)):GOTO1740
1730 A$(J,I)=SPACE$(B(J)-1)+"0"
1740 NEXTJ,I:GOTO1000
2000 IFSJ=2GOTO2250
2005 N=0:M=0:XOPEN#1,R$:INPUT#1(1),AA:IFAA=0THENCLOSE:GOTO1250
2020 FORI=1TOAA:CURSOR0,17:IFN=9GOTO2090
2030 IFN<>0GOSUB7940:GOTO2070
2040 PRINT"[ No. ]";TAB(12);"[ File Name ]";TAB(32);"[ Date ]";
2050 PRINTTAB(46);"[ No. of Data ]";TAB(63);"[ No. of Field ]":A7=1:GOSUB7080:GOSUB7940
2070 CURSOR0,17:PRINTTAB(2);N+1;".";TAB(10);A$;TAB(31);TD$;TAB(52);P3;TAB(69);NN:N=N+1
2080 A7=0:GOSUB7080:NEXT
2090 CURSOR50,17:IF(9*M+N)<AATHENPRINT"..........  To be continued.":CM=1:GOTO2110
2100 CM=0:PRINT"..........  End"
2110 CP=23:GOSUB7700:PRINTTAB(12);FS$(0):CURSOR45,23:PRINTN:CP=23:G1=1:G2=N:K=2:GF=3:GOSUB7110:ONG3+1GOTO2140,2150
2130 IFCM=1THENM=M+1:N=0:GOSUB7070:CURSOR0,17:GOTO2030
2140 CLOSE:GOTO1000
2150 I=9*M+G:GX=I:GOSUB7940:CLOSE:GOSUB7070:GOSUB7920:PRINT"reading  !!"
2180 NF=NN:N1=N2:P=P2:P1=P3:ROPEN#1,A$:FORI=0TON1:FORJ=0TOP1:INPUT#1,A$(I,J):NEXTJ,I:U=0
2200 FORI=0TON1:INPUT#1,A(I),B(I):U=U+B(I):NEXT:CLOSE:GOTO1000
2250 IFP=1GOTO1250
2252 CURSOR0,23:PRINTFF$(2);FS$(1);:G1=1:G2=2:CP=23:K=2:GF=0:GOSUB7110
2260 IFG3=0GOTO1000
2263 CURSOR12,23:PRINTSPACE$(15*(G-1));TAB(10+15*G);SPACE$(11*(3-G)):GOSUB7450:I=0:SS=G
2280 IFI=P1THENI=0:A7=0:GOSUB7080
2290 I=I+1:CURSOR0,16:PRINT"[ No.";I;" ]"
2300 FORJ=0TON1:PRINTA$(J,I);:NEXTJ:PRINT:PRINTIT$:GG=2:IFSS=2GOTO2350
2310 GF=1:GOSUB7800
2320 ONG3+1GOTO2375,1000,2350,2360
2350 CURSOR70,23:PRINT"[ STOP ]":GF=2:GOSUB7800
2355 ONG3GOTO1000,2370,2360
2360 SS=G:CURSOR12,23:IFG=2THENPRINT "                2.Step Read ":GOTO2367
2365 PRINT"1.Auto Read              "
2367 IFSS=2GOTO2350
2370 CURSOR70,23:PRINT"         "
2375 A7=2:GOSUB7080:GOTO2280
2500 IFP=1GOTO1250
2505 IFSJ=2GOTO2650
2510 ZZ=0:CURSOR 0,17:PRINT"Specify file name  (Up to 16 characters)   :"
2520 A4=0:A5=16:A6=45:GOSUB7010:IFASC(IP$)=31GOTO1000
2540 FORI=1TO16:IFASC(MID$(IP$,I,1))<>32GOTO2555
2550 NEXT:ER=1:GOSUB7400:GOTO2510
2555 GOSUB7080:GOSUB7920:PRINT"writing   !!":XX$=IP$+DT$+STR$(N1)+STR$(P)
2560 GOSUB2570:XOPEN#1,R$:INPUT#1(1),AA:AA=AA+1:PRINT#1(1),AA:PRINT#1(AA+1),XX$:CLOSE:GOTO1000
2570 WOPEN#1,IP$:FORI=0TON1:FORJ=0TOP1:PRINT#1,A$(I,J):NEXTJ,I
2580 FORI=0TON1:PRINT#1,A(I),B(I):NEXT:CLOSE:RETURN
2590 GOSUB7070:CURSOR0,17:PRINT"Should be the file on the disket deleted? ( [1]YES  /  [2]No )"
2600 A4=1:A5=1:A6=65:PI$=IP$:GOSUB7010:IFASC(IP$)=31GOTO1000
2605 IFASC(IP$)<>49GOSUB7070:GOTO2510
2610 IP$=PI$:XOPEN#1,R$:INPUT#1(GX),A$:IFIP$=LEFT$(A$,16)GOTO2625
2615 INPUT#1(1),AA:FORI=2TOAA+1:INPUT#1(I),A$:IFIP$=LEFT$(A$,16)THENGX=I:GOTO2625
2620 NEXT:CLOSE:END
2625 PRINT#1(GX),XX$:CLOSE:DELETEIP$:GOSUB2570:GOTO1000
2650 CURSOR 0,17:PRINT"Specify file name  (Up to 16 characters)   :"
2653 A4=0:A5=16:A6=45:GOSUB7010:IFASC(IP$)=31GOTO1000
2655 PRINT/PCHR$(20);CHR$(18);TAB(10);"***  Data  Processing  ***";TAB(50);DT$
2660 PRINT/P:FORI=0TON1:PRINT/PTAB(100);"Field";I+1;"= ";A$(I,0):NEXT
2665 PRINT/PCHR$(18);"    File Name :";IP$
2700 B=0:W1$="":W2$="":W3$="":W$="":W4$="          No."
```

```
2710 FORI=0TON1:W4$=W4$+"|"+SPACE$(INT(B(I)/2))+STR$(I+1)+SPACE$(B(I)-INT(B(I)/2)-1):NEXT
2720 FORI=0TON1:W$="":FORJ=1TOB(I):W$=W$+"-":NEXTJ:W1$=W1$+W$+"┬":W2$=W2$+W$+"┼":W3$=W3$+W$+"┴"
2740 NEXTI:W1$=LEFT$(W1$,LEN(W1$)-1)+"-":W2$=LEFT$(W2$,LEN(W2$)-1)+"-":W3$=LEFT$(W3$,LEN(W3$)-1)+"-":W1$="          ───┬
"+W1$:W2$=" ───┼"+W2$
2750 PRINT/PCHR$(17);W1$:PRINT/PW4$:PRINT/PW2$:W3$="          ───┴"+W3$
2770 FORI=1TOP1:PRINT/PRIGHT$("          "+STR$(I)+" ",12);"|";:FORJ=0TON1:PRINT/PA$(J,I);:IFJ<>N1THENPRINT/P"|";
2785 NEXTJ:PRINT/P:GF=1:GOSUB7800:IFG3<>0GOTO2840
2820 IFI=P1THENPRINT/PW3$:GOTO2840
2830 PRINT/PW2$:NEXTI
2840 PRINT/PCHR$(16);CHR$(21):GOTO1000
3000 IFNF=0GOTO1250
3005 CURSOR3,17:PRINT"[ Field ]";TAB(20);"[ Name ]";TAB(35);"[ Length ]";
3010 PRINTTAB(50);"[ Attribution ]":A7=2:GOSUB7080:FORI=1TO10:CURSOR0,17
3030 PRINTTAB(10-LEN(STR$(I)));STR$(I);".";TAB(16);:IFI>NFGOTO3070
3040 PRINTA$(I-1,0);TAB(40-LEN(STR$(B(I-1))));STR$(B(I-1));TAB(50);K$(A(I-1))
3070 A7=0:GOSUB7080:NEXTI:CURSOR48,17:PRINT"[No. of Store Data =";P1;" ]"
3090 CP=23:GOSUB7700:PRINTFF$(4);FS$(4):K=4:CP=23:GF=0:G1=-48:G2=207:GOSUB7110:GOTO1000
3500 IFNF=0GOTO1250
3502 IFSJ=2GOTO3600
3503 IFP=201THENER=5:GOTO1260
3505 GOSUB7450
3510 CURSOR0,16:PRINT"No.";P:CURSOR0,18:PRINTIT$
3520 FORI=0TON1:GOSUB7500
3530 IFD1=0GOTO1000
3540 NEXTI:P=P+1:P1=P-1
3550 IFP=201THENCP=23:GOSUB7700:PRINTFF$(4);ER$(5):K=4:CP=23:GF=2:GOSUB7110:GOTO1000
3560 A7=4:GOSUB7080:GOTO3510
3600 CURSOR0,17:PRINT"Specify correction item number (Within";P1;")  :"
3610 A4=1:A5=LEN(STR$(P)):A6=45:GOSUB7600
3620 IFD1=0GOTO1000
3630 IF(IP>P1)+(IP<1)GOSUB7650:GOTO3610
3640 A7=3:GOSUB7080:J=IP:GOSUB7450
3650 CURSOR0,16:PRINT"No.";IP:FORI=0TON1:PRINTA$(I,J);:NEXT:PRINT:PRINTIT$
3720 A3=0:B=P:P=J:FORI=0TON1:GOSUB7500:IFD1=0GOTO3740
3730 NEXT
3740 A3=1:P=B:A7=4:GOSUB7080:GOTO3600
4000 IFP=1GOTO1250
4005 ONSJ-1GOTO4500,4200
4007 CURSOR0,17:PRINT"Specify first deletion item number (Within";P1;") :":A5=LEN(STR$(P1))
4010 A6=50:A4=1:GOSUB7600:IFD1=0GOTO1000
4020 IF(IP<1)+(IP>P1)GOSUB7650:GOTO4010
4025 NN=IP:A7=2:GOSUB7080
4030 CURSOR0,17:PRINT"Specify number of deletion item :":A6=50:GOSUB7600
4040 IFD1=0GOTO1000
4050 IFIP<1GOSUB7650:GOTO4030
4060 A7=5:GOSUB7080:GOSUB7920:PRINT"deletion  !!"
4080 IF(NN+IP)>=PTHENP=NN:P1=P-1:GOTO4120
4090 PP=P-NN-IP:P=P-IP:P1=P-1
4100 FORJ8=0TON1:FORI8=0TOPP-1:A$(J8,NN+I8)=A$(J8,NN+IP+I8):NEXTI8,J8
4120 GOSUB7070:GOTO4000
4200 CURSOR0,23:PRINTFF$(2);FS$(2):GF=0:K=2:CP=23:G1=1:G2=2:GOSUB7110
4220 IFG3=0GOTO1000
4230 CURSOR12,23:PRINTSPACE$(15*(G-1));TAB(12+15*G);SPACE$(11*(3-G))
4240 CURSOR0,17:PRINT"Specify sort field  (within";NF;")    :":A4=1:A5=2:A6=40:GOSUB7600
4250 IFD1=0GOTO1000
4252 IF(IP<1)+(IP>NF)GOSUB7650:GOTO4240
4254 ST=IP-1:IFA(ST)=0THENER=6:GOSUB7400:CURSOR0,23:PRINTFF$(2);FS$(2):GOTO4230
4256 A7=5:GOSUB7080:GOSUB7920:PRINT"sorting  !!":Y1=0:IFP1=1GOTO1000
4300 Y2=1:Y3=P1:GOSUB4480
```

```
4310 GOSUB4490:I=Y2:J=Y3:X=VAL(A$(ST,INT(Y2+Y3)/2))
4315 IFG=1GOTO4350
4320 FORI=ITOY3:IFVAL(A$(ST,I))<XTHENNEXT
4330 FORJ=JTOY2STEP-1:IFVAL(A$(ST,J))>XTHENNEXT
4340 GOTO4370
4350 FORI=ITOY3:IFVAL(A$(ST,I))>XTHENNEXT
4360 FORJ=JTOY2STEP-1:IFVAL(A$(ST,J))<XTHENNEXT
4370 IFI>JGOTO4400
4380 FORII=0TON1:CH$=A$(II,J):A$(II,J)=A$(II,I):A$(II,I)=CH$:NEXTII
4390 I=I+1:J=J-1:GOTO4315
4400 Y4=Y3:IFY2<JTHENY3=J:GOSUB4480
4410 Y3=Y4:IFI<Y3THENY2=I:GOSUB4480
4420 IFY1<>0GOTO4310
4430 GOTO1000
4480 X(0,Y1)=Y2:X(1,Y1)=Y3:Y1=Y1+1:RETURN
4490 Y1=Y1-1:Y2=X(0,Y1):Y3=X(1,Y1):RETURN
4500 ZZ=0:X6=0:CURSOR14,17:PRINT"**** Format of Arithmetic Operation ****":A7=2:GOSUB7080
4510 FORI=0TO4:CURSOR10,17:PRINT "[";I+1;" ]   ";O$(I):A7=1:GOSUB7080:NEXT
4520 CP=23:GOSUB7700:PRINTFF$(2);FS$(3):G1=1:G2=5:GF=0:K=2:CP=23:GOSUB7110
4540 IFG3=0GOTO1000
4550 OP=G:GOSUB7070:I=0:IFG=5THENI=3
4553 GOSUB7070:CURSOR0,17:PRINTO$(G-1):A7=1:GOSUB7080
4555 A$=" OperandOperator  Result          ":GOTO4565
4557 CP=17:GOSUB7700:CURSOR21,17:PRINT"Constant Value  :":A4=1:A5=9:A6=40:GOSUB7010:IFASC(IP$)=31GOTO1000
4558 V=VAL(IP$):IFX6=1THENV1=V:DEF FNZ(I)=V1:GOTO4610
4559 DEF FNV(I)=V:GOTO4610
4565 FORI=ITO2
4568 CURSOR20,17:PRINTMID$(A$,8*I+1,8);" Field ";CHR$(65+I);" :":IFOP=5THENCURSOR35,17:PRINT" "
4570 A4=1:A5=2:A6=40:GOSUB7600:IFD1=0GOTO1000
4575 IF(I<2)*(IP<0)THENX6=X6+I+1:GOTO4557
4580 IF(IP<0)+(IP>NF)GOSUB7650:GOTO4568
4585 IFA(IP-1)=0THENER=6:GOSUB7400:GOTO4568
4590 IFOP=5THENNEXT:GOTO5020
4600 E(I)=IP-1
4610 A7=1:GOSUB7080:NEXT
4612 IFX6=0THENDEF FNZ(I)=VAL(A$(E(0),I)):DEF FNV(I)=VAL(A$(E(1),I)):GOTO4620
4614 IFX6=1THENDEF FNV(I)=VAL(A$(E(1),I)):GOTO4620
4616 IFX6=2THENDEF FNZ(I)=VAL(A$(E(0),I))
4620 A7=2:GOSUB7080:GOSUB7920:PRINT"culculation  !!":ONOP-1GOTO4650,4660,4670,5020
4640 FORI=1TOP1:Z=FNZ(I)+FNV(I):GOTO4800
4650 FORI=1TOP1:Z=FNZ(I)-FNV(I):GOTO4800
4660 FORI=1TOP1:Z=FNZ(I)*FNV(I):GOTO4800
4670 FORI=1TOP1:IFFNV(I)=0THENA$(E(2),I)=LEFT$(S$(1),B(E(2))):GOTO4820
4680 Z=FNZ(I)/FNV(I)
4800 Z=INT(Z*10+0.5)/10:A$(E(2),I)=RIGHT$(S$(0)+STR$(Z),B(E(2)))
4810 IFLEN(STR$(Z))>B(E(2))THENA$(E(2),I)=LEFT$(S$(1),B(E(2)))
4820 NEXT:GOTO1000
5020 J=IP-1:LK=0:KL=0:FORI=1TOP1:KT=VAL(A$(J,I)):LK=LK+KT:KL=KL+KT*KT:NEXT
5030 AV=INT(LK/P1*100)/100:SD=INT(100*SQR(KL/P1-AV*AV))/100
5040 GOSUB7070:CURSOR0,17:PRINT"Field";IP;" :";TAB(14);"SUM =";LK:A7=1:GOSUB7080:CURSOR14,17:PRINT"AVERAGE =";AV
5050 A7=1:GOSUB7080:CURSOR14,17:PRINT"Standard Deviation =";SD:GOTO3090
6000 IF((ERL=100)+(ERL=200))*((ERN=50)+(ERN=40))THENCURSOR0,17:PRINT"Data file does not exist.  Otherwise, disk is not
ready.":KILL:END
6005 IF(ERL=7550)+(ERL=7620)THENIP$="0":RESUME
6010 IF(ERL=2570)*(ERN=42)THENER=14:GOSUB7400:KILL:RESUME2590
6020 IF(ERN=65)+(ERN=66)+(ERN=67)THENER=ERN-57:GOTO6060
6030 IFERN=50THENER=11:GOTO6060
6040 IF(ERN=53)+(ERN=54)THENER=ERN-41:GOTO6060
6050 GOSUB7080:CURSOR0,17:PRINT"Resumption impossible":KILL:END
```

```
6060 GOSUB7420:A7=3:GOSUB7080:CURSOR0,17:IFERN=53THENPRINT"Replace with new diskette.":A7=3:GOSUB7080:CURSOR0,17
6065 PRINT"Depress any key after proper procedure."
6070 GOSUB7780:IFG=-48GOTO6070
6080 GOSUB7070:CURSOR33,17:PRINT"Resumption !!":KILL:IFERN=53GOTO6090
6082 IFERN<65THENA7=13:GOSUB7080:RESUME
6084 PRINT/P:PRINT/P:PRINT/P:PRINT/P:RESUME2655
6090 XOPEN#1,R$:INPUT#1(1),AA:CLOSE:RESUME2560
7000 A6=0:IFI=0GOTO7020
7005 FORI1=0TOI-1:A6=A6+B(I1):NEXT:GOTO7020
7010 POKE$FFFA,A4:POKE$FFFB,A5:GOTO7040
7020 POKE$FFFA,A(I):POKE$FFFB,B(I)
7040 POKE$FFFE,$50:POKE$FFFF,$D5:POKE$FFF9,A3:POKE$FFFC,A6
7050 ROPEN#1,USR($FE7A):INPUT#1,IP$:CLOSE:RETURN
7070 A7=16
7080 FORI1=0TOA7:USR($FFDD):NEXT:RETURN
7100 G1=VAL(MID$(GT$,2*JM+1,1)):G2=VAL(MID$(GT$,2*JM+2,1))
7110 R=0:RV=-1:VR=0
7120 CURSOR0,CP:IFGF=2GOTO7170
7130 GOSUB7780:IFG=-48GOTO7170
7140 IFG=-21THENG3=0:GOTO7240
7145 IFGF=3THENIFG=-16THENG3=2:GOTO7240
7150 IF(G>G1-1)*(G<G2+1)THENG3=1:GOTO7240
7160 GOSUB7650
7170 IFGF=1GOTO7120
7190 IF(GF=2)*(VR=50)GOTO7240
7200 R=0:RV=-RV:VR=VR+1:IFRV<0THENPRINTFR$:GOTO7120
7210 PRINTFF$(K):GOTO7120
7240 IFGF<>1THENPRINTFF$(K)
7250 RETURN
7400 CP=23:GOSUB7700:PRINTFF$(3);ER$(ER):GOSUB7650
7410 CP=23:K=3:GF=2:GOSUB7110:CP=23:GOSUB7700:RETURN
7420 GOSUB7070:CURSOR0,17:FR$="                    ":FF$(3)="  Error   : ":PRINTFF$(3);ER$(ER):GOSUB7650
7430 CP=17:K=3:GF=2:GOSUB7110:FF$(3)="| Error   : ":FR$="|          ":RETURN
7450 IT$="":FORII=0TON1:IT$=IT$+"^"
7455 IFB(II)>1THENFORJJ=1TOB(II)-1:IT$=IT$+"¯":NEXTJJ
7460 NEXTII:RETURN
7500 CP=23:GOSUB7700:PRINTFF$(4);"Field";I+1;": ";A$(I,0);TAB(45);K$(A(I));
7510 PRINTTAB(62);"Length=";B(I):GOSUB7000:IFASC(IP$)=31THEND1=0:RETURN
7540 IFA(I)=0GOTO7580
7550 IP$=S$(0)+STR$(VAL(IP$)):IP$=RIGHT$(IP$,B(I))
7580 A$(I,P)=IP$:D1=1:RETURN
7600 GOSUB7010:IFASC(IP$)=31THEND1=0:RETURN
7610 IFA4=0THEND1=1:RETURN
7620 IP=INT(VAL(IP$)):IP$=STR$(IP):D1=2:RETURN
7650 MUSIC"+B0R+BR+B":RETURN
7700 CURSOR0,CP:PRINTS$(SGN(CP-21)+1):CURSOR0,CP:RETURN
7780 USR($FFEA):G=PEEK($FFF9)-48:RETURN
7800 GOSUB7780:IFG=-48THENONGFGOTO7850,7800
7810 IFG=-21THENG3=1:RETURN
7820 IFG=-16THEN G3=2:RETURN
7830 IF(G>0)*(G<G6+1)THENG3=3:RETURN
7840 GOSUB7650:GOTO7800
7850 G3=0:RETURN
7920 CURSOR15,17:PRINT"Wait a minute  !!    Under ";:RETURN
7930 GOSUB7650:GOTO720
7940 INPUT#1(I+1),XX$:A$=LEFT$(XX$,16):TD$=MID$(XX$,17,12):N2=VAL(MID$(XX$,29,1)):P2=VAL(MID$(XX$,30,3)):NN=N2+1:P3=P2
-1:RETURN
```

DISK BASIC まとめ

Chapter 4

この章は、DISK BASIC MZ-2Z001の全てのコマンド、ステートメント、関数、各種オペレータを、この順にまとめています。

(GP-IBステートメントおよびRS-232Cステートメントについては、第5章および第6章を参照ください。)

4.1 DISK BASIC インタープリタMZ-2Z001のコマンド、ステートメント、ファンクション、オペレータのまとめ

4.1.1 コマンド

DIR	DIR FD*d*	ドライブ*d*番(*d*＝1～4)にあるディスクのディレクトリ (directory)を表示します。 ディレクトリ表示上に示される情報は次のものです。 (1) ディスクのボリュームナンバ (2) 使用していないセクタの総数 (3) 登録されているファイルのモード、登録状態 (LOCK されているか否か)、ファイル名 ノート：ＣＲＴ画面上のディレクトリ表示は、ファイルを１画面ぶん表示すると一旦止まり、カーソルが現れます。更に続けてディレクトリ表示を行うには CR キーを押しますが、途中で他のコマンドへ移ることもできます。
	DIR FD3	ドライブ３番にあるディスクのディレクトリを表示します。DIRコマンドを実行するとシステムは、そのドライブ番号を記憶しており、以下に示す直接実行命令、ファイルアクセス命令で、それと同一のドライブを指定する場合、ドライブ番号を省略することができます。
	DIR	一番最近DIRコマンドを実行しているドライブにあるディスクのディレクトリを表示します。
DIR/P	DIR FD2/P	ドライブ２番にあるディスクのディレクトリをラインプリンタ上に印字します。
LOAD	LOAD"A"	"A"というファイル名のついたBASICテキスト (BTX) を読み出します。
	LOAD FD2@10, "A"	ドライブ２番中の、ボリュームナンバ10番のディスクについて上記の動作を実行します。
	LIMIT $D000 : LOAD"B"	BASICテキストとリンクするための機械語プログラムファイル (OBJ) を読み出す場合は、LIMIT 命令によって BASIC エリアと機械語エリアとを分離しておく必要があります。機械語プログラムとのリンク命令を参照のこと。
LOAD/T	LOAD/T"C"	"C"というファイル名の BASIC テキストをカセットテープから読み出します。 ノート：LOADコマンドまたはLOAD/Tコマンドによって BASICテキストファイルの読み出しを実行すると、それ以前にテキストエリアにあったプログラムは無効になります。
APPEND	APPEND FD2@9, "I"	BASICテキストエリアにあるプログラムテキストと、ドライブ２番中のボリュームナンバ9番のディスクにある"I"というファイル名のついたBASICテキストを混ぜ合わせます。
SAVE	SAVE"D"	現在テキストエリアにあるBASICテキストを、"D"というファイル名を付けてディスクに書き込みます。ファイル名"D"、ファイルモードBTXのファイルが１つ登録されます。
SAVE/T	SAVE/T"E"	現在テキストエリアにあるBASICテキストを、"E"というファイル名を付けてカセットテープに書き込みます。

RUN	RUN	現在テキストエリアにあるBASICテキストの先頭からプログラムを実行します。 ノート：RUNコマンドでは、プログラムの実行直前に、すべての変数の内容を0または空（null）とします。
	RUN 1000	行番号1000からプログラムを実行します。
	RUN"F" (BTX)	"F"というBASICテキストファイルを読み出し、つづいて、テキストの先頭からプログラムを実行します。
	RUN FD3@7, "G" (OBJ)	ドライブ3番中の、ボリュームナンバ7番のディスクから機械語プログラムテキスト"G"を読み出し、つづいてその指定された実行アドレスからプログラムを実行します。この場合、システムのコントロールは、BASICから離れることになります。
VERIFY	VERIFY"H"	現在BASICテキストエリア内にあるプログラムテキストとファイル名"H"で指定するカセットテープファイルの内容を比較します。
AUTO	AUTO	テキスト作成時に、行番号を、10、20、30……と自動的に発生します。
	AUTO 200, 20	行番号を、200から20おきに、200、220、240……と自動的に発生します。 AUTO コマンドは、BREAK キーを押すことにより解除されます。
LIST	LIST	現在テキストエリア内にあるBASICテキストの全リストを表示します。
	LIST-500	行番号500までのリストを表示します。
LIST/P	LIST/P	リスト表示をラインプリンタ上へ行います。
NEW	NEW	現在テキストエリア内にあるBASICテキストを消去し、変数エリアをクリアします。LIMIT コマンドによって設定した機械語プログラムエリアはクリアされません。
CONT	CONT	プログラム実行を継続します。即ちプログラム中のSTOPステートメントあるいは、BREAK キーによって中断された箇所から実行を再開します。 プログラムの中断時に、BASICテキストのエディションを行うとCONTコマンドは無効になります。
MON	MON	システムのコントロールをBASICからモニタへ移します。(モニタからBASICへの復帰は、モニタコマンド"J"によって行うことができます。)
BOOT	BOOT	MZ-2000のIPL を起動して新たにシステムソフトウェアをローディングします。
KLIST	KLIST	デファイナブル・ファンクション・キーの定義状態を調べるため、各機能をCRTディスプレイ上にリストします。

4.1.2 ファイルコントロール文

LOCK	LOCK"ABC"	最後にDIR FD*d*コマンドを実行しているドライブ(アクティブ・ドライブ)中のファイル"ABC"をロックします。
	LOCK FD4@7, "ABC"	ドライブ4番中の、ボリュームナンバ7番のディスク上のファイル"ABC"をロックします。 ロックされたファイルは、変更または削除を受け付けません。ディレクトリ表示では、ロックされたファイルに*記号が付けて示されます。
UNLOCK	UNLOCK"ABC"	アクティブドライブ中のファイル"ABC"のロックを解除します。
	100 UNLOCK FD1, "A"	ドライブ1番中のディスク上のファイル"A"のロック解除をプログラム上で実行します。

RENAME	RENAME"A" , "B"	アクティブドライブ中のファイル"A"について、ファイル名を"B"に変更します。
DELETE	DELETE"A"	アクティブドライブ中のファイル"A"をディスク上から削除します。
CHAIN	CHAIN FD2@7 , "TEXT B"	ドライブ2番中にあるボリュームナンバ7番のディスク上にあるBASIC テキスト"TEXT B"にプログラム実行をチェインします。即ち、"TEXT B"をBASIC テキストエリアに読み出し、その先頭からプログラム実行を続けます。 このとき、テキストエリア中にあったもとのプログラムはNEWされますが、変数の値や利用者関数の内容は、CHAIN されるテキストに受け渡されます。CHAIN 文の働きは、GOTO "file name"として理解することができます。
	CHAIN"TEXT B"	プログラムを、アクティブドライブ中のBTXファイル"TEXT B"にチェインします。
SWAP	SWAP FD2@7, "TEXT S-R"	ドライブ2番中にあるボリュームナンバ7番のディスク上にあるBASICテキスト"TEXT S-R"にプログラム実行をスワップします。 即ち、実行中のテキストをアクティブドライブ中のディスク上に一旦待避させ、次に、"TEXT S-R"をBASICテキストエリアに読み出し、その先頭からプログラム実行を続けます。 スワップされたプログラムが終了したら、今度は、もとのテキストを読み出して、SWAP ステートメントの次のステートメントからプログラム実行を続けます。各プログラム実行のリンクの際には、変数の値や、利用者関数の内容は、受け渡されます。SWAP レベルは1を越えてはなりません。即ち、SWAPされたテキストで更にSWAPインストラクションを行うことはできません。 SWAP文の働きは、GOSUB "file name"として理解することができます。

4.1.3 BSD（BASIC シーケンシャルアクセス・データファイル）コントロール文

WOPEN#	WOPEN #3 , FD2@7 , "SEQ DATA1 "	1つの BASIC シーケンシャルアクセスデータファイル（BSD）を作成するために書き込み用ファイルをオープンします。即ち，作成する BSD のファイル名を"SEQ DATA1"と定義し、ドライブ2番中にあるボリュームナンバ7番のディスク上に、ロジカルナンバ3番としてファイルをオープンします。 USR関数に対するWOPEN # ステートメントは、P.86に示されています。
PRINT#	PRINT#3 , A , A$	WOPEN#ステートメントによってロジカルナンバ3番にファイルオープンされている BSD に、変数A、ストリング変数A$ の内容を順に書き込みます。 BSDは、CLOSE# ステートメントによってファイルクローズが実行されてはじめて1つの BSD として正式に登録されます。
CLOSE#	CLOSE#3 (corresponding to WOPEN#)	WOPEN# ステートメントでロジカルナンバ3番にファイルオープンされたBSD をクローズします。 ファイルクローズによって、WOPEN#ステートメントによって定義されたファイル名をもつ1つのBSD が指定ディスク上に作成され、ロジカルナンバ（この場合3番）は未定義のものに戻します。
KILL#	KILL#3	WOPEN#ステートメントでロジカルナンバ3番にファイルオープンしたBSDをキルします。即ち、BSDの作成をキャンセルし、ロジカルナンバ（この場合3番）を未定義のものに戻します。
ROPEN#	ROPEN #4 , FD2@7 , "SEQ DATA 1"	BASICシーケンシャルアクセスデータファイル（BSD）中のデータを読み出すためにファイルをオープンします。即ち、ドライブ2番中のボリュームナンバ7番のディスク上にあるBSDファイル"SEQ DATA 1"を、ロジカルナンバ4番としてファイルオープンします。 USR関数に対するROPEN# ステートメントは、P.86に示されています。

INPUT#	INPUT#4 , A(1) , B$	ROPEN#ステートメントによってロジカルナンバ4番にファイルオープンされているBSDから、順次データを読み出し、配列要素A(1)に数値データを、ストリング変数B$にストリングを代入します。 読み出すデータは、BSDの先頭データから順次シーケンシャルにアクセスされます。
CLOSE#	CLOSE#4 (corresponding to ROPEN#)	ROPEN#ステートメントで、ロジカルナンバ3番にファイルオープンされたBSDをクローズします。 ロジカルナンバ4番は未定義のものに戻します。

4.1.4 BRD (BASICランダムアクセス・データファイル) コントロール文

XOPEN#	XOPEN #5 , FD3@18 , "DATA R1"	BASICランダムアクセスデータファイル(BRD)へのデータ書き込み/読み出しのためファイルオープンをします。(cross open) 即ち、ドライブ3番ボリュームナンバ18番のディスク上にあるBRDファイル"DATA R1"をロジカルナンバ5番にクロスオープンするか、或いは、まだ存在しないファイルなのであれば、新たにそのディスク上にBRDファイル"DATA R1"を作成するためにクロスオープンします。
PRINT#( )	PRINT#5(11) , R(11)	XOPEN#ステートメントによってロジカルナンバ5番にファイルオープンされているBRDの要素11に、1次元数値配列要素R(11)の内容を書き込みます。
	PRINT#5(20) , AR$, AS$	上記と同じBRDの要素20、要素21にそれぞれ、ストリング変数AR$、AS$の内容を書き込みます。BRD中の要素は全て32バイトの固定長であるので、ストリング長が32バイトを越えると、越えた分は無効となります。
INPUT#( )	INPUT#5(21) , R$	XOPEN#ステートメントによってロジカルナンバ5番にファイルオープンされているBRDの要素21にあるデータをストリング変数R$に読み出します(代入します)。
	INPUT#5(11) , A(11) , A$(12)	上記と同じBRDの要素11、要素12にあるデータをそれぞれ、1次元数値配列要素A(11)、1次元ストリング配列要素A$(12)に読み出します。
CLOSE#	CLOSE#5	XOPEN#ステートメントでロジカルナンバ5番にファイルオープンされたBRDをクローズします。
	CLOSE	WOPEN、ROPENあるいはXOPENされているすべてのファイルをクローズします。
KILL#	KILL	WOPEN、ROPENあるいはXOPENされているすべてのファイルをキルします。
IF EOF(#)	IF EOF(#5) THEN 700	BSDに対してINPUT#ステートメントを実行した時、あるいはBRDに対してINPUT#( )ステートメントを実行した時、もしファイルエンドが発生したら、行番号700の処理ルーチンへジャンプせよ、という分岐文です。

4.1.5 エラー処理文

ON ERROR GOTO	ON ERROR GOTO 1000	プログラム実行中にエラーが発生したら、行番号1000にジャンプせよという宣言文です。
IF ERN	IF ERN=44 THEN 1050	エラー番号が44であれば行番号1050へジャンプせよという命令です。
IF ERL	IF ERL=350 THEN 1090	エラー発生行番号が350であれば行番号1090へジャンプせよという命令です。
	IF (ERN=53)*(ERL=700) THEN END	エラー番号が53で、かつエラー発生行番号が700であるならば、プログラムを終了せよという命令です。 DISK BASICでは、プログラム中でエラーが発生したら、変数ERN、ERLにそれぞれ、エラー番号、エラー発生行番号がセットされます。

RESUME		エラー処理後、メインプログラムへ復帰する命令です。復帰のし方によって次のようなそれぞれの使い方ができます。
	650 RESUME	エラーが発生した命令へ再びコントロールを移します。
	700 RESUME NEXT	エラーが発生した命令の次の命令へコントロールを移します。
	750 RESUME 400	行番号400へコントロールを移します。
	800 RESUME 0	プログラムの先頭へコントロールを移します。

4.1.6 カセットテープ・データファイル入出力文

WOPEN/T	10 WOPEN/T " DATA-1 "	カセットテープ・データファイル" DATA-1 " を書き込みオープンします。
PRINT/T	20 PRINT/T A , A$	数値変数A、ストリング変数A$ の内容をWOPEN/Tによってオープンされているカセットテープデータファイルに書き込みます。
CLOSE/T	30 CLOSE/T	WOPEN/T によってオープンしたファイルをクローズします。
ROPEN/T	110 ROPEN/T " DATA-2 "	カセットテープ・データファイル" DATA-2 " を読み出しオープンします。
INPUT/T	120 INPUT/T B , B$	ROPEN/T によってオープンしたカセットテープ・データファイル中のデータを順に読み出し、数値変数B、ストリング変数B$ に代入します。
CLOSE/T	130 CLOSE/T	ROPEN/T によってオープンしたファイルをクローズします。

4.1.7 代入文

LET	〈LET〉 A=X+3	数値変数Xと数値データ3の加算結果を数値変数Aに代入します。LET は省略できます。

4.1.8 入出力文

PRINT	10 PRINT A	CRTディスプレイ上に数値変数Aの内容を表示します。
	?A$	CRTディスプレイ上にストリング変数A$の内容を表示します。
	100 PRINT A ; A$, B ; B$	数値変数とストリング変数を混合して使用できます。また区切りでセミコロンが使われると、スペースなしで続けて表示され、コンマが使われると次の表示位置(10文字ごとの区切り)から表示されます。
	110 PRINT " COST= " ; CS	クォーテーションマーク「"」で囲まれたストリングはその内容がそのまま表示されます。
	120 PRINT	PRINT だけの場合は、行替えになります。
INPUT	10 INPUT A	キーボードから変数Aに対する数値を入力します。
	20 INPUT A$	キーボードからストリング変数A$ に対するストリングを入力します。
	30 INPUT " VALUE? " ; D	キーボードから入力する前に、ストリングの VALUE? を表示させます。ストリングと変数の区切りはセミコロン「;」を使います。
	40 INPUT X , X$, Y , Y$	数値変数やストリング変数はコンマ「,」で区切れば混合して使用できますが、入力する際には変数の型に合わせる必要があります。

GET	10 GET N	キーボードから数値変数Nに対して、1文字の数値を入力します。そのときキーが押されていないと、0が入力されます。
	20 GET K$	キーボードからストリング変数K$に対して、1個のストリングを入力します。そのときキーが押されていないと、A$は空になります。
READ～DATA		DATA文に置かれた定数またはストリングをREAD文の中に示された変数に代入する命令です。READ文中の変数と、それに対応するDATA文中の各データは、数値変数なら数値データ、ストリング変数ならストリングデータと、変数とデータの形が一致しなくてはなりません。
	10 READ A，B，C 1010 DATA 25，−0.5，500	左のREAD～DATA文の実行によって数値変数A、B、Cのそれぞれに数値データ25、−0.5、500が代入されます。
	10 READ H$，H，S$，S 30 DATA HEART，3 35 DATA SPADE，11	READ文の最初の変数、すなわちストリング変数H$にDATA文の最初のデータ、すなわちストリングデータ"HEART"が代入されます。次に2番目の変数Hには、数値データ3が代入され、次々に読み込まれて行きます。
RESTORE		READ…DATA命令では、READ命令につれてDATA文中から読み込むデータが移って行きますが、RESTORE文を使うことによって読み込むデータをテキスト中のDATA文の最初に戻すことができます。
	10 READ A，B，C 20 RESTORE 30 READ D，E 100 DATA 3，6，9，12，15	左の例では、行番号10のREAD文によって、変数A、B、Cにそれぞれ、3、6、9の値が代入されますが、次にRESTORE文が置かれているので、行番号30のREADによって変数D、Eに代入される値は、12、15とはならずに、それぞれ3、6が代入されることになります。
	700 RESTORE 200	READ～DATA文におけるデータ読み出しポインタを、行番号200のDATA文の先頭へ移します。

4.1.9 ループ文

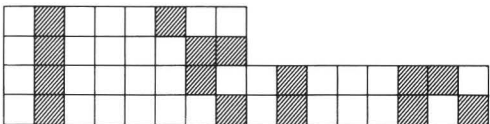
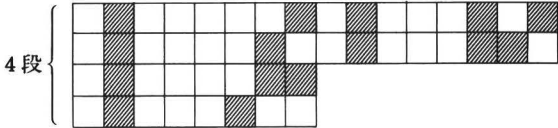
FOR～TO **NEXT**	10 FOR A=1 TO 10 20 PRINT A 30 NEXT A	行番号10は変数Aを1から10まで変化させよという命令で、最初Aの値は1となります。行番号20でAの値がCRTディスプレイに表示されるので数値1が表示されます。次に行番号30でAの値は2になってこのループを繰り返します。こうしてAの値が10になるまでこのループが繰り返されます。(ループを終了した時点でAには11の値が入っています。)
	10 FOR B=2 TO 8 STEP 3 20 PRINT B^2 30 NEXT	行番号10は変数Bを2から8まで、3ずつ大きくして変化させよという命令です。STEPの値を負にして変数の値を小さくして行くこともできます。
	10 FOR A=1 TO 3 ─┐ 20 FOR B=10 TO 30 ┐ │ Bのループ Aのループ 30 PRINT A，B 40 NEXT B ─┘ 50 NEXT A ─┘	変数AとBについてFOR……NEXTループを重ねた例です。BループはAループの内部に置かれているところに注目して下さい。ループは2重、3重……とネスティングすることができますが、内側のループは必ず外側のループ内に閉じていなくてはなりません。 FOR……NEXTのネスティングは15レベルを越えてはなりません。
	60 NEXT B，A 70 NEXT A，B	前の2重ループで、行番号40と50を、左の行番号60のように1つのNEXT文にまとめることができます。しかし行番号70に示したようなオペランドではエラーになります。

4.1.10 分岐文

GOTO	100 GOTO 200	行番号200へジャンプ（=プログラム実行を移すこと）します。
GOSUB **～RETURN**	100 GOSUB 700 ……………… 800 RETURN	行番号700のサブルーチンへ分岐（=サブルーチンをコールすること）します。RETURN文でサブルーチンの実行を終了し、メインプログラムでGOSUB命令をした次のステートメントへ戻ります。

IF～THEN	10 IF A＞20 THEN 200	変数Aの値が20より大なら、行番号200へジャンプします。Aが20以下なら次の行を実行します。
	50 IF B＜3 THEN B＝B＋3	変数Bの値が3より小なら変数BにB＋3の値を代入します。Bが3以上なら次の行を実行します。
IF～GOTO	100 IF A＞＝B THEN 10	変数Aの値が変数Bの値以上なら行番号10へジャンプします。AがBより小なら次の行を実行します。
IF～GOSUB	30 IF A＝B＊2 GOSUB 90	変数Aの値が変数Bの値の2倍に等しいなら行番号90のサブルーチンへ分岐します。等しくないなら次の行を実行します。 (条件文のあとにマルチステートメントが来る場合、ON 文は条件が成り立たないとき実行されますが、IF 文は条件が成り立たないとき次の行番号へ移りマルチステートメントは無視されます。)
ON～GOTO	50 ON A GOTO 70，80，90	変数Aの値が1なら行番号70へ、2なら行番号80へ、3なら行番号90へジャンプします。Aが0または4以上なら次の文を実行します。ONにはINTの機能が含まれておりAが2.7ならば2の場合と同様に行番号80へジャンプします。
ON～GOSUB	90 ON A GOSUB 700，800	変数Aの値が1なら行番号700、2なら行番号800のサブルーチンへ分岐します。Aが0または3以上なら次の文を実行します。

4.1.11 定義文

DIM		配列を使う場合には、このDIM (dimension の略) 文で、配列要素の最大を宣言しておかなければなりません。配列要素は0から、最大255まで使えます。
	10 DIM A(20)	1次元数値配列A( )について、配列要素をA(0)からA(20)まで21個用意します。
	20 DIM B(79，79)	2次元数値配列B( )について、配列要素をB(0, 0)からB(79, 79)まで6400個用意します。
	30 DIM C1$(10)	1次元ストリング配列C1$( )について、配列要素をC1$(0)からC1$(10)まで11個用意します。
	40 DIM K$(7，5)	2次元ストリング配列K$( )について、配列要素を、K$(0，0)からK$(7，5)まで48個用意します。
DEF FN	100 DEF FNA(X)＝X^2－X 110 DEF FNB(X)＝LOG(X)＋1 120 DEF FNZ(Y)＝LN(Y)	DEF FNで関数の定義をします。行番号100はX^2-XをFNA(X)に、文110は$\log_{10}X+1$をFNB(X)に、文120は$\log_e Y$をFNZ(Y)に定義します。関数は1変数に限ります。
DEF KEY	15 DEF KEY(1)＝LIST⏎ 25 DEF KEY(2)＝LOAD!RUN⏎	DEF KEY文は、デファイナブル・ファンクションキーの機能定義を行います。行番号15の DEF KEY 文は、ファンクションキー1番に、LIST [CR] の機能を定義し、行番号25では、LOAD：RUN [CR] の機能が定義されます。

4.1.12 注釈文とコントロール文

REM	200 REM JOB-1	REMは注釈文であり、プログラム実行の際無視されます。
STOP	850 STOP	プログラムの実行をやめて、命令待ちとなります。ここでCONT命令を与えると、プログラムを続行します。
END	1999 END	プログラムの最後を表わします。プログラムの実行をやめますが、CONT命令を与えると、さらに先のプログラムを実行します。

CLR	300 CLR	数値をとる変数および配列要素の内容をすべて0、ストリングをとる変数および配列要素の内容をすべて空（null）とします。
CURSOR	50 CURSOR 25 , 15 60 PRINT "ABC"	CRTディスプレイの任意の箇所にカーソルを移動させるのがCURSORです。X軸方向には、40キャラクタモードのときは0～39、80キャラクタモードのときは0～79、Y軸方向には上端から0～24の数値または変数で位置を指定します。左の例では、ストリング"ＡＢＣ"は、画面の左端から26番目、上端から16番目のカーソル位置から表示されます。
CSRH		現在のカーソル位置のX座標（水平位置）を示すシステム数値変数です。
CSRV		現在のカーソル位置のY座標（垂直位置）を示すシステム数値変数です。
CONSOLE	10 CONSOLE S10 , 20	CRTディスプレイ上のスクロールエリアを第10行から20行までとします。
	20 CONSOLE C80	CRTディスプレイ上のキャラクタ表示を80キャラクタ／行にします。
	30 CONSOLE C40	CRTディスプレイ上のキャラクタ表示を40キャラクタ／行にします。
	40 CONSOLE R	CRTディスプレイをリバース表示とします。
	50 CONSOLE N	CRTディスプレイをノーマル表示に戻します。
	60 CONSOLE GN	CRTディスプレイ上のグラフィック表示のリゾリューションモードを320×200ドット／画面にします。
	70 CONSOLE GH	CRTディスプレイ上のグラフィック表示のリゾリューションモードを640×200ドット／画面にします。
CHANGE	10 CHANGE	メインキーボード上の26個のアルファベットキーのシフトポジション大文字／小文字を逆にします。
REW	710 REW	カセットテープを巻き戻します。
FAST	720 FAST	カセットテープを早送りします。
SIZE	? SIZE	BASIC テキストエリアの未使用バイトサイズを表示させます。
TI$	100 TI$＝"222030"	内蔵時計を午後10時20分30秒に合わせます。時刻データはクォーテーションマーク内に6桁の数字で表現します。

4.1.13 ミュージックコントロール文

MUSIC **TEMPO**		音楽の自動演奏をさせる命令です。文字通りTEMPO文で指定されたテンポにのって、MUSIC文にあるクォーテーション「"」内の旋律のストリング（音程と音長の指定された音符の集まりに相当します）を音にしてスピーカを鳴らします。
	300 TEMPO 7 310 MUSIC "DE#FGA"	行番号300でテンポが7（最も速い速度です）に指定されます。行番号310で、7のテンポで中音域のレミファの#ソラが続けて演奏されます。TEMPO文が無かったらテンポはデフォルト値で演奏されます。
	300 M1$＝"C3EG＋C" 310 M2$＝"BGD－G" 320 M3$＝"C8R5" 330 MUSIC M1$, M2$, M3$	この例では、旋律を3つのストリング変数に代入してMUSIC命令を実行させるものです。五線符で示すと下の音符が演奏されます。なお、TEMPO文がありませんので、デフォルト値のテンポで演奏されます。

4.1.14 グラフィックコントロール文

GRAPH	10 GRAPH I1	グラフィックエリアへのデータ転送モードを、ページ1（グラフィックエリア1）に設定します。
	20 GRAPH O1	グラフィックエリア1を表示モードとします。
	30 GRAPH O23	グラフィックエリア2および3を表示モードとします。
	40 GRAPH O123	グラフィックエリア1、2および3を表示モードとします。
	50 GRAPH O0	グラフィックエリアの表示を行わないようにします。
	60 GRAPH C	GRAPH I 文によってデータ転送モードとなっているエリアをクリアします。
	70 GRAPH F	転送モードとなっているグラフィックエリアをフィル（fill）します。
	80 GRAPH I1 , C , O1	3つの命令を行うものではじめに、データ転送をグラフィックエリア1とし、続いてエリア1をクリアして、表示モードもエリア1とします。
SET		グラフィックエリアの任意の位置にドットを表示します。 オペランドはX座標、Y座標の順に指定します。 X方向は画面の左から右へ0～319(ハイリゾリューションモードの場合は0～639)、Y方向は画面の上から下へ0～199の範囲で指定します。
	300 SET 160 , 100	画面の中央にドットをセットします。(320×200ドット/画面の場合)
RESET		SET 文の逆で、オペランドで指定した位置を Black ドットとします。
	310 RESET 160 , 100	画面の中央のドットをリセットします。(320×200ドット/画面の場合)
LINE		SET 文の拡張で、オペランドで指定するドット位置を次々に直線で結びます。
	400 LINE 110 , 50 , 210 , 50 , 210 , 150 , 110 , 150 , 110 , 50	画面の中央に1辺の長さ100の正方形を描きます。(320×200ドット/画面の場合)
BLINE		LINE文の逆で、オペランドで指定する直線をBlackラインとします。
POSITION		グラフィックエリア上のポジションポインタを設定します。次の PATTERN 文は、このポジションポインタの位置から実行されます。
	20 GRAPH I2 , C , O2 30 POSITION 0 , 50 40 PATTERN 8 , A$	行番号20で、グラフィックエリア2を転送モード、また表示モードとして、エリアをクリアし、行番号30、40で座標位置（0 ,50）からグラフィックパターンを表示します。グラフィックパターンは下から上方へ8段重ねで表示されます。
PATTERN		POSITION 文で指定されたポジションポインタ位置から、任意のドットパターンを8ビット単位で表示します。ドットパターンはストリングデータ又は、ストリング変数で与えます。オペランドの第一項は、8ビット単位のドットパターンを上下どちらの方向に何段組にするか決めます。
	10 C$ = " ABCDEF " 20 PATTERN 4 , C$	次のドットパターンが表示されます。 4段

	30 PATTERN −4 , C$	次のドットパターンを表示します。 4段
POINT	100 ON POINT (X , Y) GOTO 10 , 20 , 30 , 40 , 50 , 60 , 70	グラフィックエリア上の座標点（X , Y）がセットされているかリセットされているかで分岐する文です。 **POINT関数の関数値　　　　　ポイント情報** 0 ……………… グラフィックエリア1 , 2および3ともリセットされている。 1 ……………… グラフィックエリア1のみセットされている。 2 ……………… グラフィックエリア2のみセットされている。 3 ……………… グラフィックエリア1,2のみセットされている。 4 ……………… グラフィックエリア3のみセットされている。 5 ……………… グラフィックエリア1,3のみセットされている。 6 ……………… グラフィックエリア2,3のみセットされている。 7 ……………… グラフィックエリア1,2,3すべてがセットされている。 （注：上記情報はグラフィックエリア1、2、3すべてが装備されている場合のものであり、いずれかのエリアが、装備されていない場合については、BASIC/MONITOR MANUALを参照ください。）
POSH		グラフィック表示エリア上のポジションポインタの現在のX座標を示すシステム変数。 POSHのとり得る値は 0≦POSH≦319（ノーマルリゾリューションモード） 0≦POSH≦639（ハイリゾリューションモード） となります。
POSV		グラフィック表示エリア上のポジションポインタの現在のY座標を示すシステム変数。

4.1.15　機械語プログラムコントロール文

LIMIT	100 LIMIT 49151	BASICプログラムで使用するエリアを、49151番地（16進でBFFF）に制限します。
	100 LIMIT A	BASIC プログラムで使用するエリアを、変数Aの値の番地に制限します。
	100 LIMIT $BFFF	BASIC プログラムで使用するエリアを、16進番地 BFFF に制限します。16進表現する場合は、このように" $ "マークを用います。
	300 LIMIT MAX	BASIC プログラムで使用するエリアを、メモリの最大に戻します。
	200 LIMIT $BFFF 210 LOAD FD2 , "S-R1"	機械語プログラムファイル（OBJ）" S-R1 "が、ローディングアドレスC000 以上のものであれば、左のプログラムによって、機械語リンクエリア内に、" S-R1 "が、ドライブ2番中のディスケットより読み出されます。
POKE	120 POKE 49450 , 175	10進番地49450にデータ175（10進表現）をセットします。
	130 POKE AD , DA	変数ADで指定する番地に、変数DAで示される値（0～255の範囲）をセットします。
PEEK	150 A=PEEK(49450)	10進番地49450にはいっているデータを10進数に直して変数Aに代入します。
	160 B=PEEK(C)	変数Cで指定される10進番地にはいっているデータを10進数に直して変数Bに代入します。

USR	500 USR(49152)	10進番地49152にプログラムのコントロールを移します。このコントロールの移動は、機械語の CALL コマンドと同じ機能を持っています。従って、機械語プログラムに、RETコマンド、(10進コードで201) があると、BASIC プログラムへリターンします。
	550 USR(AD)	変数ADで指定される10進番地を CALL します。
	570 USR($C000)	16進番地C000をCALLします。
	600 WOPEN#8, USR ($C000) 610 PRINT#8, A$ 620 CLOSE#8	USR ($C000) を、ロジカルナンバ8番に定義して書き込みオープンします。行番号610で、ストリング変数A$の内容がセットされているバッファの先頭番地をDEレジスタにセットし、データの長さ (最大255バイト) をBCレジスタにセットして、USR ($C000) を実行します。
	700 ROPEN#9, USR ($C100) 710 INPUT#9, B$ 720 CLOSE#9	USR ($C100) を、ロジカルナンバ9番に定義して読み出しオープンします。行番号710でUSR ($C100) が実行されますが、コールされた機械語ルーチンでコールされた時のDEレジスタの示すアドレスを先頭として、ストリングデータをセットし、その長さをBCレジスタにセットしてリターンすると、B$にそのストリングデータが代入されます。

4.1.16 プリンタ・コントロール文

PRINT/P		PRINTと同様の機能を、オプションのプリンタに対して実行します。プリンタが接続されていない時はエラーになります。
	10 PRINT/P A, A$	数値変数Aの内容、続けてストリング変数A$の内容をプリンタにプリントします。
	20 PRINT/P CHR$(5)	プリンタのフォーム・フィードを行います。(CHR$(5)はプリンタコントロールコードです。)
IMAGE/P	30 IMAGE/P CHR$(255), "UU"	プリンタに任意のイメージドットパターンをプリントします。
COPY/P	10 COPY/P 1	プリンタにCRTディスプレイ上のキャラクタ表示のコピーをとります。
	20 COPY/P 2	プリンタにCRTディスプレイ上のグラフィックエリア1のコピーをとります。
	30 COPY/P 3	プリンタにCRTディスプレイ上のグラフィックエリア2のコピーをとります。
	40 COPY/P 4	プリンタにCRTディスプレイ上のグラフィックエリア1と2のコピーをとります。
	50 COPY/P 5	プリンタにCRTディスプレイ上のグラフィックエリア3のコピーをとります。
	60 COPY/P 6	プリンタにCRTディスプレイ上のグラフィックエリア1と3のコピーをとります。
	70 COPY/P 7	プリンタにCRTディスプレイ上のグラフィックエリア2と3のコピーをとります。
	80 COPY/P 8	プリンタにCRTディスプレイ上のグラフィックエリア1、2および3のコピーをとります。
PAGE/P	100 PAGE/P 20	プリンタの1ページを20行に設定します。

4.1.17 I/O 入出力文

INP		I/Oポート番号を指定して、そのポート上にあるデータの読み出しを行います。
	10 INP @12 , A 20 PRINT A	行番号10で、I/Oポート番号12（10進）にあるデータを、変数Aに読み出します。
OUT		外部デバイスにデータを送るため、I/Oポート番号を指定してデータを出力します。
	30 B＝A^2＋0.3 40 OUT @13 , B	行番号40で、変数Bの値をI/Oポート番号13の出力ポートへ出力します。

4.1.18 数値関数

ABS	100 A＝ABS(X)	変数Xの値の絶対値｜X｜を変数Aに代入します。カッコ内は定数、数式でもかまいません。 (例) ABS (−3)＝3 ABS (12)＝12
INT	100 A＝INT(X)	変数Xの値について、Xを越えない最大の整数を求めて変数Aに代入します。カッコ内は定数、数式でもかまいません。 (例) INT (3 . 87)＝3 INT (0 . 6)＝0 INT (−3 . 87)＝−4
SGN	100 A＝SGN(X)	変数Xの値についてX＜0のとき−1を、X＝0のとき0を、X＞0のとき、1を変数Aに代入します。カッコ内は定数、数式でもかまいません。 (例) SGN (0 . 4)＝1 SGN (0)＝0 SGN (−400)＝−1
SQR	100 A＝SQR(X)	変数Xの値について、$\sqrt{X}$の値を求めて変数Aに代入します。カッコ内は定数、数式でもかまいませんが、正または0の値でなければなりません。
SIN	100 A＝SIN(X)	変数Xの値（ラジアン）について、sinXの値を求め変数Aに代入します。カッコ内は定数、数式でもかまいません。ラジアンと度の関係は、 1度＝$\frac{\pi}{180}$ラジアン
	110 A＝SIN(30＊π/180)	ですから、たとえばsin30°の値を変数Aに代入するには行番号110のようにします。
COS	200 A＝COS(X) 210 A＝COS(200＊π/180)	変数Xの値（ラジアン）について、cosXの値を求め変数Aに代入します。カッコ内は定数、数式でもかまいません。度で計算するにはSIN関数と同様の方法を使います。行番号 210 は cos200°の値を変数Aに代入する命令文です。
TAN	300 A＝TAN(X) 310 A＝TAN(Y＊π/180)	変数Xの値（ラジアン）について、tanXの値を求め変数Aに代入します。カッコ内は定数、数式でもかまいません。度で計算するにはSIN関数と同様の方法を使います。行番号 310 はtanY°の値を変数Aに代入する命令文です。
ATN	400 X＝ATN(A) 410 Y＝180/π＊ATN(A)	変数Xの値について、$\tan^{-1}X$の値（ラジアン）を求め変数Aに代入します。カッコ内は定数、数式でもかまいません。計算結果は$-\frac{\pi}{2}$と$\frac{\pi}{2}$の間の値となります。行番号410は$\tan^{-1}X$の値を度にして変数Aに代入する命令文です。
EXP	100 A＝EXP(X)	変数Xの値について、e^Xの値を求めて変数Aに代入します。カッコ内は定数、数式でもかまいません。
LOG	100 A＝LOG(X)	変数Xの値について、常用対数$\log_{10}X$の値を求めて変数Aに代入します。カッコ内は定数、数式でもかまいませんが、正の値でなければなりません。

LN	100 A＝LN(X)	変数Xの値について、自然対数$\log_e$Xの値を求めて変数Aに代入します。カッコ内は定数、数式でもかまいませんが、正の値でなければなりません。
	110 A＝LOG(X)/LOG(Y) 120 A＝LN(X)/LN(Y)	対数の底がYのときの対数$\log_Y$Xを求めるには行番号110または行番号120によって求められます。
RND		0.00000001から0.99999999までの値をとる擬似乱数を発生する関数です。カッコ内に0または負の整数を与える場合と、正の整数を与える場合とで、2通りの処理が行われます。
	100 A＝RND(1) 110 B＝RND(10)	行番号100または110のようにカッコ内に正の整数を与えるとRND関数を使うたびに、順次0.00000001から0.99999999までの間の値をとる乱数値を発生します。(カッコ内に与える正の整数の値には無関係です。)
	200 A＝RND(0) 210 B＝RND(－3)	行番号200または210のようにカッコ内に0または負の整数を与えると乱数発生のイニシャライズが行なわれて、いつもある特定の数値を発生してAにもBにも同じ値が代入されます。

4.1.19 ストリングコントロール関数

LEFT $	10 A$＝LEFT$(X$，N)	ストリング変数X$の最初からN文字目までを、ストリング変数A$に代入します。Nは定数でも、変数、数式でもかまいません。
MID $	20 B$＝MID$(X$，M，N)	ストリング変数X$の第M文字目からN文字を、ストリング変数B$に代入します。
RIGHT $	30 C$＝RIGHT $(X$，N)	ストリング変数X$の右からN文字を、ストリング変数C$に代入します。
SPACE $	40 D$＝SPACE $(N)	N個のスペースをストリング変数D$に代入します。
STRING $	50 E$＝STRING $("＊", 10)	10個の連続したアステリスクマークを、ストリング変数E$に代入します。
CHR $	60 F$＝CHR $(A)	ASC関数の逆で、変数Aの値に等しいASCIIコードの文字（キャラクタ）をストリング変数F$に代入します。Aは定数、変数、数式いずれでもかまいません。
ASC	70 A＝ASC(X$)	ストリング変数X$の最初の文字のASCIIコード（10進数）の値を、変数Aに代入します。
STR$	80 N$＝STR$(I)	VAL関数の逆で、変数Iの数値をそのままストリングとして、ストリング変数N$に代入します。
VAL	90 I＝VAL(N$)	ストリング変数N$の数字ストリングを、そのまま数値として変数Iに代入します。
CHARACTER$	85 CR$＝CHARACTER$(X，Y)	キャラクタ表示位置（X，Y）に現在表示されているキャラクタをストリング変数CR$に代入します。
LEN	100 LX＝LEN(X$)	ストリング変数X$の文字の長さ（文字数）を、変数LXに代入します。
	110 LS＝LEN(X$＋Y$)	ストリング変数X$とY$の文字の長さの和を、変数LSに代入します。

4.1.20　タブ関数

TAB	10 PRINT TAB(X)；A	画面の左端から数えてX＋1字目に変数Aの値を表示します。

4.1.21　算術演算子

左端の白ぬきの数字は計算の優先順位、更に優先されるのはカッコ(　)内の計算です。

❶^	10 A＝X^Y(べき乗)	変数AにX^{Y}の計算結果を代入します。 (但しX^YでXが負数のとき、Yが整数でなければエラーとなります。)
❷−	10 A＝−B(負号)	0−Bは減算ですが、−Bの「−」は負号であることに注意して下さい。
❸＊	10 A＝X＊Y(乗算)	変数AにXとYの数値の乗算結果を代入します。
❸/	10 A＝X/Y(除算)	変数AにXとYの数値の除算結果を代入します。
❹＋	10 A＝X＋Y(加算)	変数AにXとYの数値の加算結果を代入します。
❹−	10 A＝X−Y(減算)	変数AにXとYの数値の減算結果を代入します。

4.1.22　比較・論理演算子

＝	10 IF A＝X THEN…	変数AとXの数値が等しいならば、THEN 以降の命令を実行します。
	20 IF A$＝"XYZ" THEN…	ストリング変数A$の内容がストリングXYZであれば、THEN 以降の命令を実行します。
＞	10 IF A＞X THEN…	変数AがXより大きいならば、THEN以降の命令を実行します。
＜	10 IF A＜X THEN…	変数AがXより小さいならば、THEN以降の命令を実行します。
＜＞or＞＜	10 IF A＜＞X THEN…	変数AとXの数値が等しくないならば、THEN 以降の命令を実行します。
＞＝or＝＞	10 IF A＞＝X THEN…	変数AがXより大きいか等しいならば、THEN 以降の命令を実行します。
＜＝or＝＜	10 IF A＜＝X THEN…	変数AがXより小さいか等しいならば、THEN 以降の命令を実行します。
＊	40 IF(A＞X)＊(B＞Y) THEN…	変数AがXより大きく、かつ変数BがYより大きいならば、THEN 以降の命令を実行します。
＋	50 IF(A＞X)＋(B＞Y) THEN…	変数AがXより大きいか、または変数BがYより大きいならば、THEN 以降の命令を実行します。

4.1.23　その他のシンボル

?	200 ? "A＋B＝"；A＋B 210 PRINT"A＋B＝"；A＋B	PRINTの代わりに用いることができます。したがって行番号200と210は同等です。
:	220 A＝X：B＝X^2：?A，B	命令文の区切りを表わす記号で、多重命令に使用します。行番号220の多重命令には、3つの命令文が置かれています。
;	230 PRINT"AB"；"CD"；"EF"	PRINTを続けて実行します。行番号230では画面上に、「ABCDEF」とスペースを空けずに続けて表示されます。
	240 INPUT"X＝"；X$	画面に「X＝」と表示し、ストリング変数X$のデータキー入力を待ちます。
,	250 PRINT"AB"，"CD"，"E"	タブレーションをつけてPRINTを実行します。行番号250の文では、画面上にまずABと表示し、次にAから10文字右の場所よりCDと表示し、次にCから10文字右の場所にEと表示されます。

	300 DIM A(20), B$(3, 6)	変数の区切りに用いた例です。
" "	320 A$=" SHARP BASIC " 330 B$=" MZ-2000 "	" "内がストリングであることを示します。
$	340 C$="ABC"+CHR$(3)	ストリング変数であることを示します。
	500 LIMIT $BFFF	16進数であることを示します。
π	550 S=SIN(X*π/180)	円周率の近似値3.1415927をπで表わします。

GP-IBステートメント

Chapter
5

本章ではGP-IBインタフェースカードMZ-8BI04を用いて、計測器等をコントロールする場合のBASICステートメントに関する内容を含んでいます。
GP-IBに関する詳細は、GP-IBインタフェースの解説書を参照ください。

なお、GP-IBインタフェースの解説書は、MZ-80B用に作成されていますが、MZ-2000用としても使用できます。ただし、スレーブモードにおけるサンプルプログラム例の中で、BASICとリンクするための番地が定義されていますが、BASICインタープリタMZ-2Z001（バージョンV1.0a）を使用の際は、下記の通り、リンキングアドレスおよびインタラプトテーブルのアドレスを変更する必要がありますので、注意願います。

GP-IB INTERFACE解説書
P.84（リンキングアドレスの変更）

P.85（インタラプトテーブルのアドレス変更）

ステートメントの書式

■ステートメントを、アルファベット小文字、反転文字で表記することはできません。

■かぎカッコ〈 〉中の要素は、省略あるいは、任意回の繰り返し記述が可能な要素であることを示しています。実際のプログラムにこのカッコをタイプしてはいけません。

■セパレータ（コロン、セミコロンなど）は、決められた位置に正しく置かなくてはなりません。

BASICを使用すると、大部分のGP-IBに特有なコマンドを意識することなく、プログラムの作成が可能です。しかし、規格で定められたコマンドには、コマンドを送出する順序が決められており、この順序を誤ると、計測器は動作しません。また、計測器には、それぞれアドレスがつけられており、このアドレスは1つのシステム内に同じアドレスを持つ計測器が2台以上存在すると、不適当な動作をします。
さらに、制御される側の計測器についても、それぞれ、動作に対応したコードが決められており、このコードは同じメーカの同じ計測器以外は、すべて異なっています。従って、プログラムを作成する場合には、測定器の取扱い説明書を良く読み、理解しておくことが必要です。また、データの終了を示すコードについても同様で、EOIだけのものもあれば、EOIをださずASCIIコードで代用するもの、この2つの併用型等、各種あります。これらはすべて各々の計測器に個有のものであり、規格の定める範囲外にあり、また、BASICコマンドにも含まれずデータとして送るストリングデータに含まれますので、これらのことを考えながらプログラムを作成する必要があります。

GP-IBステートメントリスト

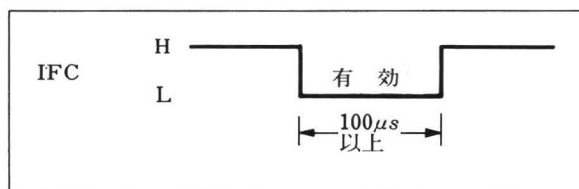
ステートメント	内　　容
1．ICL	MZ-2000がシステムコントローラとして、インタフェース全体を初期状態に設定。
2．REN	MZ-2000がシステムコントローラとして、インタフェース全体をリモート・モードに設定します。
3．LCL	MZ-2000がシステムコントローラとして、インタフェース全体をローカル・モードに設定します。
4．LCL *n*	MZ-2000がコントローラとして、インタフェース・バス上の *n* で指定される機器をローカルに戻します。
5．LLO	MZ-2000がコントローラとして、インタフェース・バス上の機器すべてにローカルに戻ることを禁止します。
6．DCL	MZ-2000がコントローラとして、インタフェース・バス上の機器すべてを、機器側で定められている初期状態に設定します。
7．DCL *n*	MZ-2000がコントローラとして、インタフェース・バス上の *n* で指定される機器を機器で定められている初期状態に設定します。
8．TRG *n*	MZ-2000がコントローラとして、インタフェース・バス上の *n* で指定される機器を動作開始させます。
9．PCT *n*	MZ-2000がコントローラとして、バス上の他のコントローラ機能を持つ機器にコントローラの権利を渡します。その後はコントローラとしての動作はエラー。
10．WRT *n, v*	MZ-2000はコントローラとして、*n* で指定される機器をリスナに、トーカを自分に設定し、*v* で指定するデータを送出します。
11．RED *n, v*	MZ-2000はコントローラとして、*n* で指定される機器をトーカに、リスナに自分を設定し、*v* に受信データを格納します。
12．WRT/ *v*	MZ-2000がトーカとして、*v* で指定するデータを送出します。
13．RED/ *v*	MZ-2000がリスナとして、*v* で指定される変数に、受信データを格納。
14．CMDW *x*$	MZ-2000がコントローラとして *x*$ で指定するコマンドを送出し、その後MZ-2000はトーカとなります。
15．CMDR *x*$	MZ-2000がコントローラとして *x*$ で指定するコマンドを送出し、その後MZ-2000はリスナとなります。
16．ON SRQ	バス上の機器のどれかがサービス要求をしている場合、その処理ルーチンにジャンプ。
17．SPOL *n, v*	バス上の機器 *n* に対して、MZ-2000がコントローラとして、シリアル・ポールを実行し、ステータスバイトを変数 *v* に格納します。
18．PPC *n, l*	MZ-2000がパラレル・ポールを実行する時に、バス上の機器が応答するステータスビットを *l* で、機器アドレスを *n* で指定します。(10進で指定)
19．PPOL *v*	MZ-2000がコントローラとしてパラレルポールを実行し、ステータスビットを変数 *v* に格納します。
20．PPU	MZ-2000がパラレル・ポールモードを解除します。
21．GPIBM *n*	MZ-2000が常にシステムコントローラであり、アクティブコントローラであって、自分のアドレスを *n* で指定します。
22．EOIW *n*	データ送出時のデリミタ指定。
23．EOIR *n*	データ受信時のデリミタ指定。

5.1 ステートメント

MZ-2000がBASICを起動したとき、システムコントローラ、アクティブコントローラ、アドレスNo. 1に設定されますが、I/O動作は何もしていません。

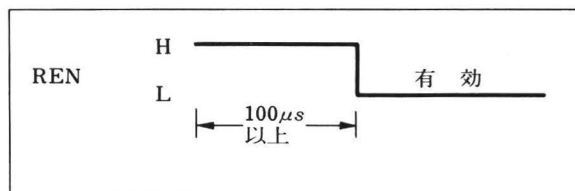
5.1.1. ICL (Interface Clear)

書　式　ICL

機　能　IFCラインに100μs以上の負のパルスを発生させ、インタフェース全体を初期設定します。

注　意
- MZ-2000がシステムコントローラでないときは、エラーとなります。
- 変数は不要です。

5.1.2 REN (Remote Enable)

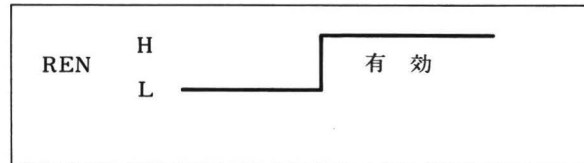
書　式　REN

機　能　RENラインを100μs間Highレベルにした後、RENラインをLowレベルに設定し、バス上の機器をリモートモードに設定します。

注　意
- MZ-2000がシステムコントローラでないときはエラーとなります。
- 変数は不要です。

5.1.3 LCL (Local)

書 式 LCL

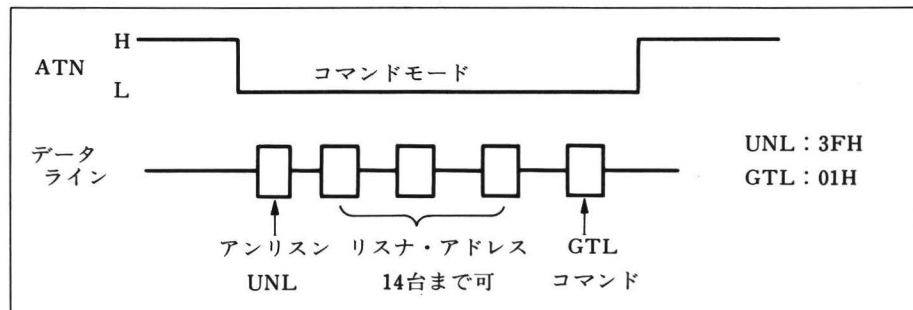
機 能 RENラインをHighレベルに設定し、バス上の機器をすべてローカルモード（手動状態）に戻します。

注 意

- MZ-2000がシステムコントローラでないときはエラーとなります。
- 変数は不要です。
- 再度、測定器を制御するときにはRENコマンドの実行が必要です。

5.1.4 LCL *n* (Go to local No.*n*)

書 式 LCL n_1 <; n_2 ……; n_k>

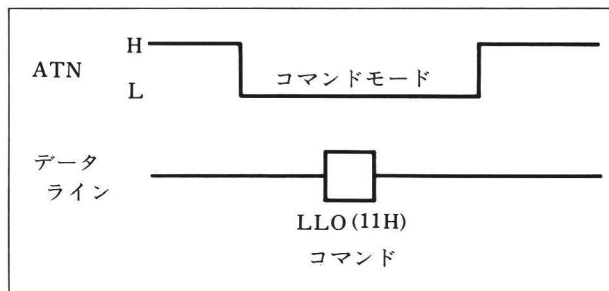
n_i：装置番号（リスナ）

機 能 MZ-2000はATNラインをLowレベルに設定し（コマンドモード）、リスナを解除し、n_iで指定された機器のリスナアドレスを送出した後にGTLコマンドを送り、指定されたリスナをローカルモードにします。

注 意

- n_iは０～30までの10進数での指定が可能です。
- 指定可能なリスナの台数は14以内。
- MZ-2000がアクティブコントローラでないときはエラーとなります。

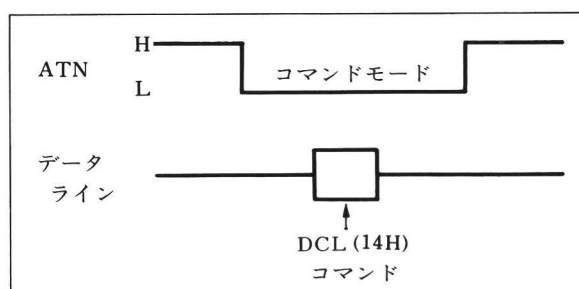
5.1.5 LLO (Local Lock-out)

書　式　LLO

機　能　MZ-2000はバスをコマンドモードにし、LLOコマンドを送り、バス上の機器すべてに対してローカルモードに戻ることを禁止します。(すなわち、フロントパネル操作を無効にさせます。)

注　意　●MZ-2000がアクティブコントローラでないときはエラーになります。
●変数は不要です。

5.1.6 DCL (Device Clear)

書　式　DCL

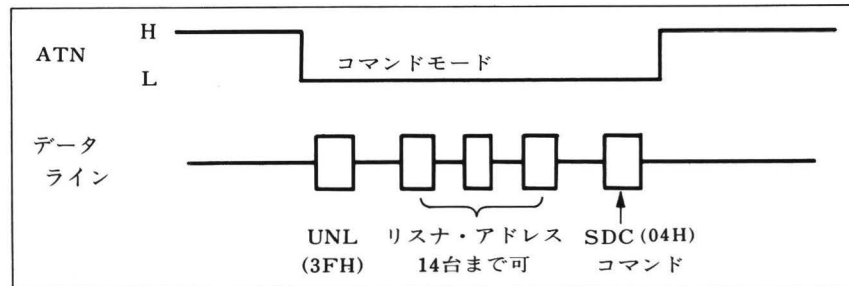
機　能　MZ-2000はバスをコマンドモードにし、DCLコマンドを送出し、バス上の機器すべてを機器個有の初期状態に設定します。

注　意　●MZ-2000がアクティブコントローラでないときはエラーとなります。
●変数は不要です。
●機器個有の初期状態は各計測器のマニュアルを参照ください。

5.1.7 DCL *n* (Selected Device Clear)

書　式　DCL n_1 <; n_2 ;…… ; n_k>

n_i：装置番号（リスナ）

機　能　MZ-2000はバスをコマンドモードにし、UNLコードによってすべてのリスナを解除した後、n_iで指定される機器のリスナアドレスを送り、続けてSDCコマンドを送出し、指定したリスナを機器個有の初期状態に設定します。

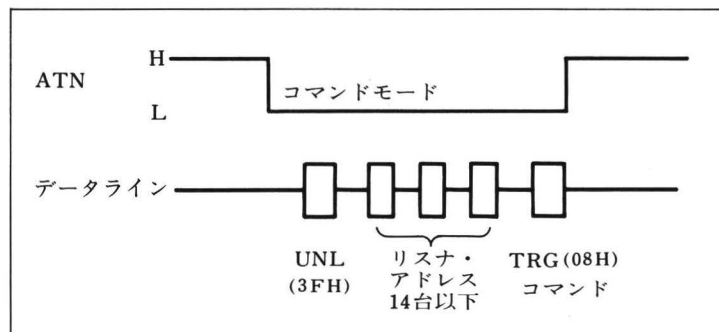
指定台分のリスナが初期設定されます。

注　意

- n_iは0～30までの10進数で指定します。
- 指定可能なリスナ台数は14以下です。
- MZ-2000がアクティブコントローラでないときはエラーとなります。

5.1.8 TRG (Trigger)

書　式　TRG n_1<; n_2 ; …… ; n_k>

n_i：装置番号（リスナ）

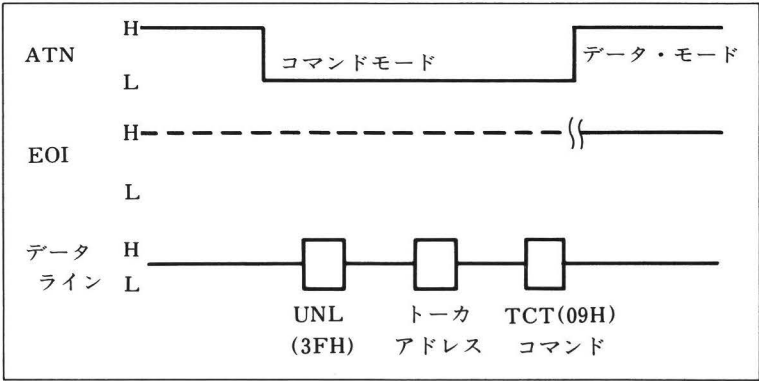
機　能　MZ-2000はバスをコマンドモードにし、リスナ解除した後n_iで指定される機器のリスナアドレスに続けてTRGコマンドを送り、指定した台分のリスナを動作開始（測定開始）させます。

注　意

- n_iは0～30までの10進数が使用可能です。
- 指定するリスナの台数は14以下です。
- MZ-2000がアクティブコントローラでないときはエラーとなります。

5.1.9 PCT (Pass Control)

書式 PCT *n*

n：装置番号（トーカ）

機能 MZ-2000はバスをコマンドモードにし、リスナを解除した後、コントローラ機能をもつ*n*で指定される機器のトーカアドレスに続けてTCTコマンドを送り、その機器にコントローラの権利を渡す。

注意

- *n*はコントローラ機能をもつ機器の番号であり、指定個数は1。
- コマンドモードではMZ-2000はアクティブコントローラですが、データ・モードになったときバス上で単なるリスナまたはトーカに指定されるまでコマンドは送出できません。ICL実行の場合に、アクティブなコントローラに復帰できます。

5.1.10 WRT (Write)

書式

WRT n_1〈; n_2 ; …… ; n_k〉, v_1〈v_2, ……, v_m〉

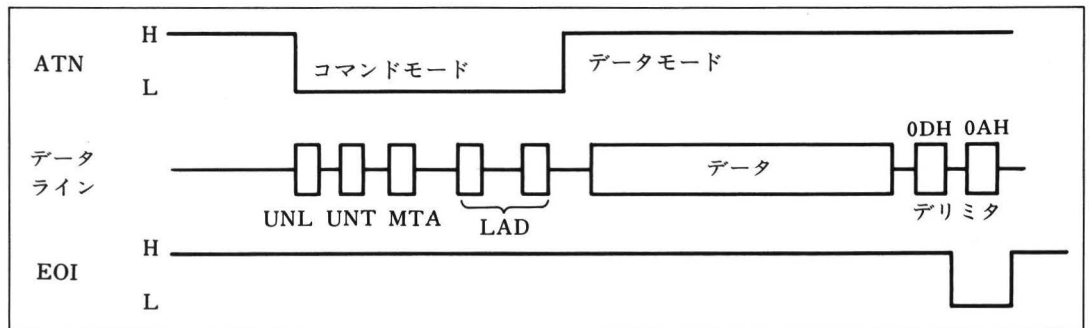
n_i：装置番号（リスナ）

v_j：転送データ（ストリング変数、数値変数または定数）

機能

指定したリスナに対して指定したデータを転送します。

MZ-2000はコントローラとして、コマンドモードにおいて自分をトーカに指定し、n_iで示される機器をリスナに指定します。

コマンド送出後、MZ-2000はバスをデータモードにし、リスナにv_jで指定されるデータを送出します。データ送出後、自動的にデリミタコード0DH-0AHおよびEOIを下図のタイミングで送出します。

注意

●転送データとして数値を用いた場合、ストリングに変換されてASCIIコードで出力されます。転送データが複数個指定された場合は一連のASCIIコードとして連続して出力されます。

●装置番号n_iは０～30までの10進数が有効です。

●指定可能なリスナ台数は14以下です。

●コントローラインチャージ（能動になれるコントローラ）の場合に実行可。
　パスコントロール後に実行するとエラーとなります。

例

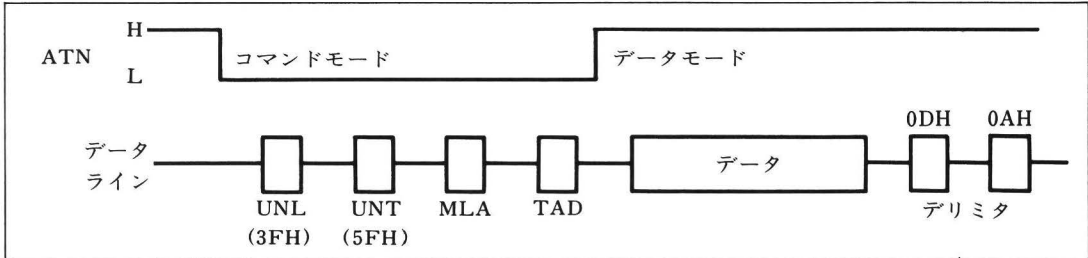
```
10 WRT 5, "SP1;" ………………………… 5番の装置に "SP1;" が送られる。

10 MZ＝3 ：PC＝4 ：D$＝ "ABC"
20 WRT MZ;PC, D$…………………… 3番、4番の装置に "ABC" を送ります。

10 PL＝5 ：PN＝4
20 WRT PL, "SP", PN, ";" ………… 5番の装置に "SP4;" を送ることになります。
```

5.1.11 RED (Read)

書式 RED n, v_1 〈, v_2, ……, v_k〉

n：装置番号（トーカ）

v_i：受信データ格納のための変数

機能 指定のトーカからデータを受信し、指定の変数に格納します。MZ-2000はコマンドモードにおいて自分をリスナに指定し、nで示される装置をトーカに指定し、データモードにおいてトーカから送られるデータを入力します。EOIラインは1バイトごとにチェックし、EOIがLowレベルであれば、そのときのデータライン上のデータを受信して終了します。しかしEOIがHighレベルであればデリミタコード(0DH-0AH)を受信したときにデータ受信を終了します。

注意

- 格納変数としてストリング変数を使用した場合、デリミタまでのデータ（ただし253文字以内）を格納するため、ストリング変数は1つのみ使用できます。
- 数値変数を用いた場合、一連のデータのうち、コンマ(,)またはデリミタまでを数値に変換して各々の変数に順次格納します。(変換できない場合はエラーとなります。)
- 複数の数値変数を用いた場合、データの個数と変数の個数が一致している必要があります。
- 測定器から送られてくるデータのフォーマットは統一されていないので、この命令を使ってデータを入力する場合、受信データのフォーマットに注意する必要があります。

例

10 RED 5, A$ ……………………………… 5番の装置からデータを入力し、A$に格納する。

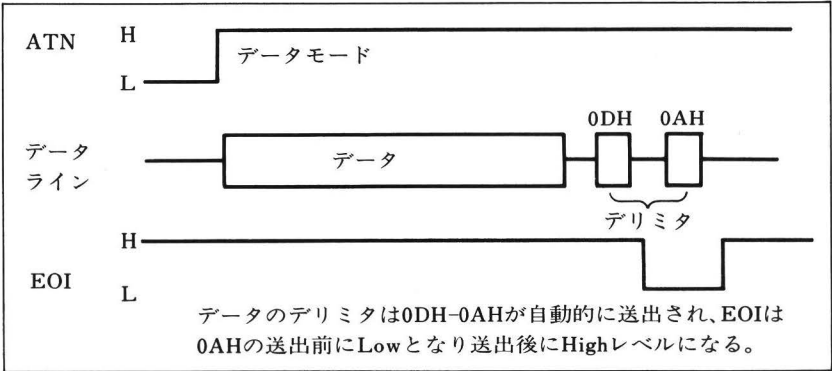
10 FC＝24
20 RED FC, F1, F2 ……………………… 送られてきたデータが100, 240とすればF1＝100, F2＝240となる。

5.1.12 WRT/

書　式　WRT/v_1〈, v_2, …, v_k〉

v_i：転送データ

機　能　自分がトーカに指定されることを確認し、指定されていればデータを送出します。指定されていなければエラーとなります。

転送データのフォーマットに関してはWRT *n*, *v*と同様です。同一の装置に多数のデータを送る場合、一度トーカ、リスナを指定しておけば、そのトーカ，リスナは解除あるいは再指定まで有効ですから、WRT/命令によりデータモードのみでデータの転送ができます。

例

```
10 WRT 9, A$
20 FOR I=1 TO 100
30 WRT/A$
40 NEXT I
```

9番の装置にA$の内容を101回送る。

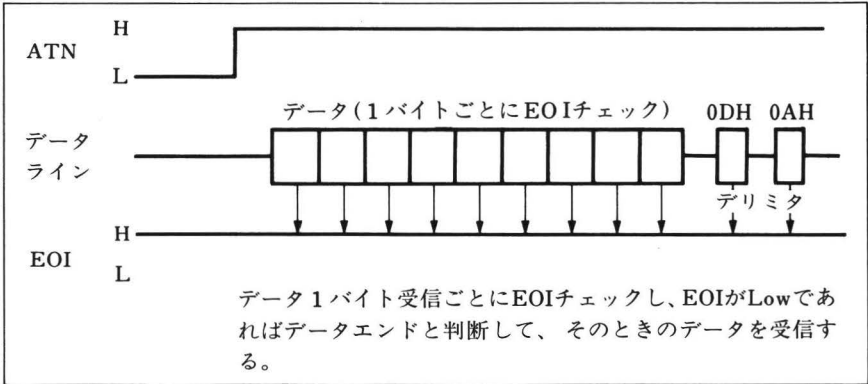
5.1.13 RED/

書　式　RED/v_1〈, v_2, …, v_k〉

v_i：データ格納のための変数

機　能　自分がリスナに指定されていることを確認し、指定されていればデータを入力し、変数に格納します。リスナに指定されていなければエラーとなります。

データ格納変数に関しては、RED n, vの場合と同様です。

WRT/命令の場合と同様に、同一のトーカからのデータを何回も受信する場合に、この命令は有効です。

例

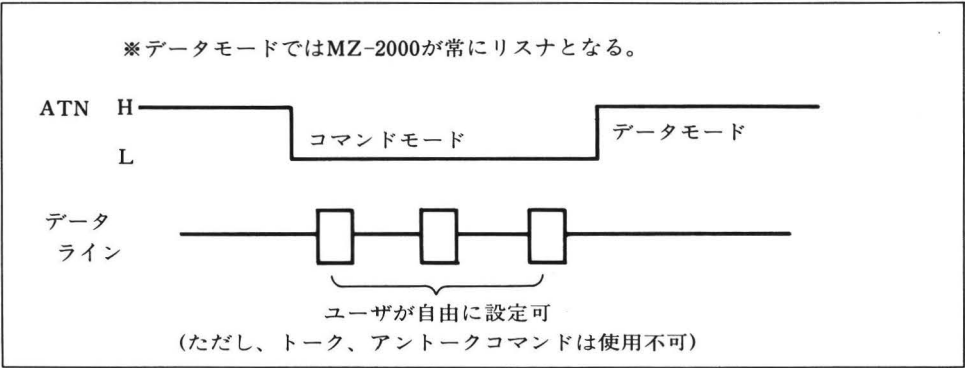
```
10 DIM A$(100)
20 RED 13, A$(0)
30 FOR I=1 TO 100
40 RED/A$(I)
50 NEXT 1
```

13番の装置からデータを101回入力し、A$( )に入れる。

5.1.14 CMDW (Command Write)

書式　CMDW *x*$

x$：転送データ

機能　MZ-2000はコマンドモードにおいて自分自身をトーカに指定し、*x*$で示されるコマンドに対応するストリングをバスに送出した後、データモードに設定する。

注意

- *x*$中にはUNT(5FH)、TAD(40H〜5EH)は使用しないでください。
- 同じリスナと同じトーカ間で何回もデータを転送するときに、このステートメントを使用すると便利です。コマンドの設定が1回ですみます。
- 2次アドレスを持つ機器のアドレスが可能です。
- このステートメントはユーザが自由に使用できますが、規格(IEC,IEEE-488)の内容を理解しないで使用すると全く動作しない場合があります。

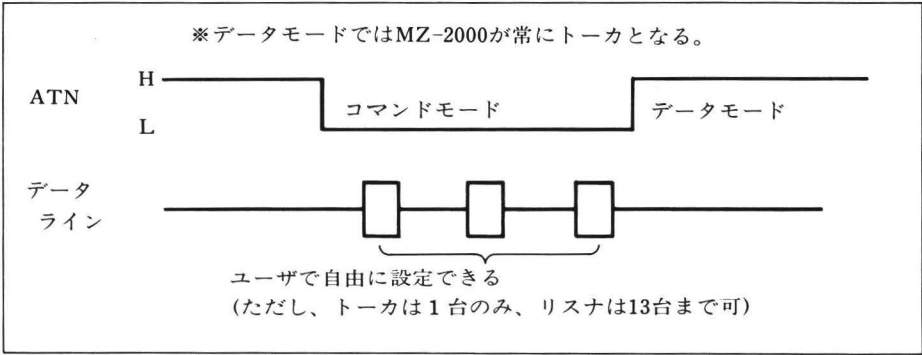
例

```
10 A$ =CHR$ ($3F)+CHR$ ($25) ···························UNL、およびプロッタをリスナ
20 B$ = "LBAAA" +CHR$ ($03)+CHR$ ($3B) ·····················デリミタ+デリミタ
30 CMDW A$ ··········································トーカMZ-2000、リスナプロッタに指定
40 FOR I=0 TO 100
50 WRT/B$ ···············································································101回データを書く
60 NEXT
```

5.1.15 CMDR (Command Read)

書　式　CMDR *x*$

x$ ：受信データ

機　能　MZ-2000はコマンドモードにおいて自分自身をリスナに指定し、*x*$で指定するコマンドに対応するストリングをバスに送出した後、データモードに設定する。

注　意

- MZ-2000がリスナに指定されているので、トーカを1台指定すれば、ユーザが指定できるリスナ台数は13台までとなります。
- 同じリスナ、同じトーカ間でデータを何回も転送する際のコマンド指定にこのステートメントを使用すると便利です。
- 2次アドレスの送出に使用できます。
- このステートメントはユーザが自由に使用できますが、規格(IEC,IEEE-488)の内容を理解しないで使用すると全く動作しない場合があります。

例

```
10 A$=CHR$($3F)+CHR$($5F)+CHR$($51)
             ↑          ↑          ↑
            UNL        UNT     トークアドレス

20 CMDR A$ ……………… MZ-2000をリスナ、HP-3325A(F.G)をトーカに指定
30 RED/B$ ……………………………… トーカからのデータをB$に格納
40 PRINT B$
```

5.1.16 ON SRQ

書　式　ON SRQ l_n

l_n：行番号

機　能　このステートメントを最初に実行しておくと、バス上の機器が、サービスを要求してきたときにlnで示される行番号からはじまる処理ルーチンへジャンプします。

注　意　このステートメントは1回のみ有効であり、処理ルーチンの実行完了後、再度必要な場合には改めてこのステートメントを設定しなければなりません。

5.1.17 SPOL (Serial Polling)

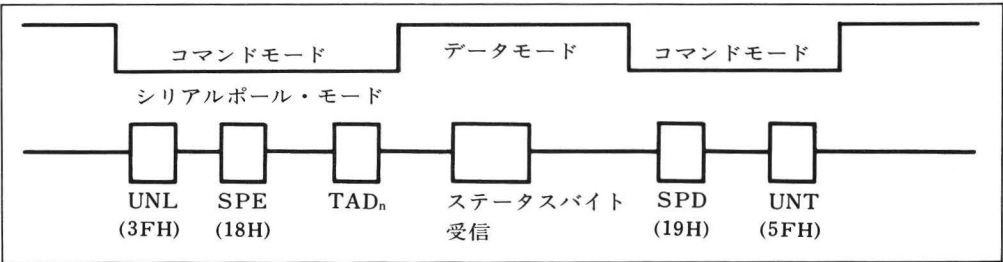
書　式　SPOL n, v

n：装置番号（トーカ）

v：ステータスバイト

機　能　MZ-2000はコントローラとしてリスナ解除し、SPEコマンドにつづけてnで指定される機器のトーカアドレスを送り、データモードにします。

データモードにおいて指定されたトーカからのステータスバイトを受信し変数vに格納します。

再度コマンドモードに設定し、SPEコマンド、UNTコマンドを送り、シリアルポールモードを解除します。

注　意　指定トーカ1台のみで、nで示されるアドレスは0～30までの10進数が有効です。

例

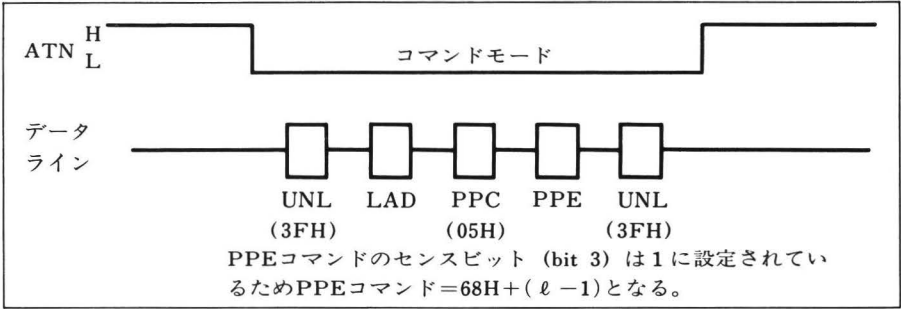
```
10 ON SRQ 150 : N=5
20 ICL : REN : DCL
30 WRT 5, "IM252, 32, 16;" …………エラーが生じたときSRQを発生させる。
40 WRT 5, "SP1; SP2; SP9;" ……….SP9でエラーが発生。
50 WRT 5, "IN;" : GOTO 900 ………エラーランプ消去。
150 SPOL N, A …………………………Nで指定されるトーカのステータスを変数Aに格納。
160 IF A>=64 THEN 50…………………Aが40H以上であれば要求源確認。
170 N= N+1 : GOTO 150 ………………次のトーカをチェックする。
900 END
```

5.1.18 PPC (Parallel Poll Configure)

書　式　PPC *n*, *l*

n：装置番号（リスナ）

l：データライン番号

機　能　MZ-2000はコントローラとしてバスをコマンドモードにしリスナを解除し、*n*で指定される機器のリスナアドレスを送り、続けてPPCコマンドを送出し、さらに（*l* − 1）+68Hで示されるPPEコマンドを送出し、パラレルポール時にリスナに指定した機器が応答するデータラインを割り振ります。

注　意

- PPCステートメントはプログラムでPPOLを実行する以前に実行しておくことが必要です。
- PPEコマンドのセンスビットを0(ゼロ)にする場合は、CMDWまたはCMDRを用いてコマンドを上図の手順で(PPEのみ60H+(*l* − 1)として)送出します。
- *n*は0〜30までの10進数が有効です。
- *l*は1〜8までの10進数が有効で、応答するデータライン番号を示します。
- このステートメントはPP機能のサブセットがPP1のものに対してのみ有効で、PP2（ハードウェアで応答ビットが決定されている)を使用しているものに関しては無効です。
- PPEコマンドの解除はPPD(70H)コマンドを送出しなければなりません（CMDWまたはCMDRを用いること。)

5.1.19 PPOL (Parallel Poll)

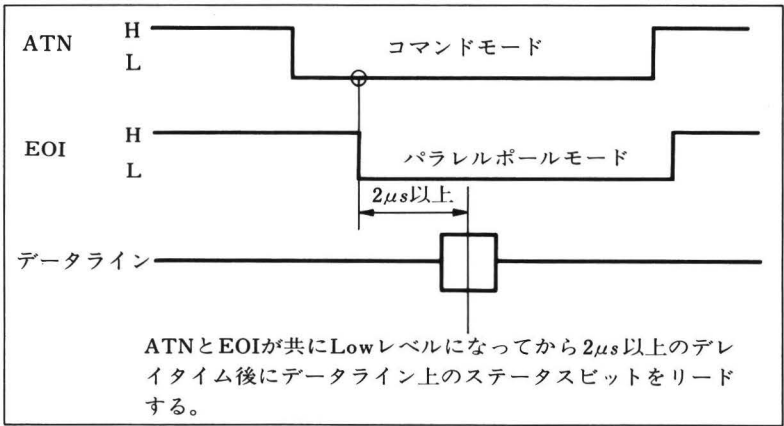
書 式

PPOL *v*

v ：ステータスビット格納変数

機 能

MZ-2000はコントローラとして任意の時点でATNとEOIラインを共にLowレベルに設定してパラレルポールを実行します。

この時のみ3線ハンドシェークを使用しないでリスナがステータスビットを設定する時間（約2 μs）だけ待った後、データライン上のビットパターンを変数 *v* に格納します。

注 意

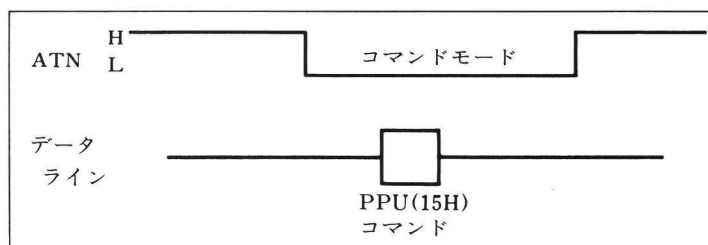
変数 *v* は10進数で 0 ～255までの範囲です。

例

```
10 ON SRQ 100
20 ICL : REN : DCL
30 WRT 5, "IM252, 32, 16;" ……………エラーが生じたらSRQを発生する。
40 WRT 5, "SP1; SP2; SP7;" …………SP7でエラーが生じ、SRQが発生したので。
50 WRT 5, "SP0; IN;" : GOTO 900 …プロッタを初期設定して終了。
100 PPOL A ……………………………………パラレルポール実行。
110 IF A=4 THEN 50 ………………………A=4ならOK。
120 PRINT "ERROR" ……………………A≠4ならエラー表示。
900 END
```

5.1.20 PPU (Parallel Poll Unconfigure)

書　式　PPU

機　能　MZ-2000はコマンドモードにおいてPPUコマンドを送出しパラレルポールモードを解除します。

注　意　パラレルポールを再実行するためには再度PPC*n*,*l* のステートメントを実行しなければなりません。

5.1.21 GPIBM

書　式　GPIBM *n*

n ：装置番号

機　能　MZ-2000をシステムコントローラ、アクティブコントローラに設定し、*n* で指定される 0 ～30までの10進数を自分の装置番号とします。

5.1.22 EOIW

書　式　EOIW n_1　または　EOIW n_1, n_2

n_1 n_2：デリミタ

機　能　データ送出時のデリミタとして 2 個までの10進数で指定します。

注　意
- 3 個以上のデリミタは無視します。
- EOIは最後のデリミタの送出前後の間だけLowレベルになります。

5.1.23 EOIR

書　式　EOIR n_1　または　EOIR n_1, n_2

n_1、n_2：デリミタ

機　能　データ受信時のデリミタとして 2 個までの10進数で指定します。

注　意
- 3 個以上のデリミタは無視します。
- EOIラインがLowレベルであれば、そのときのデータライン上のデータを受信後、終了します。

5.2 プログラム作成上の注意

アドレス

各計測器、機器にはそれぞれ個有のアドレスが設定されており、このアドレスはかえられるようになっています。

1つのシステム内に同じアドレスを持つ計測器があると誤動作の原因になりますので注意が必要です。

データのデリミタ

各計測器には、データの終了を判断するためのコード、すなわちデリミタが設定されていますが、これは各種類あり、極端な場合、同一測定器であっても動作モードによって、デリミタが異なる場合があります。

MZ-2000では初期設定として、0DH-0AHの2バイトがデリミタとして定義されていますが、使用する計測器のデリミタに合せる必要があります。

さもなければ、データ転送が途中で停止し、動作しなくなります。

デリミタは計測器の取扱い説明書に示してありますから、それを参照してください。

ステートメントの実行順序

シリアルポール時の "ON SRQ" パラレルポールの "PPC *n*, *l*" アドレスの設定、デリミタの指定、ICL、REN等はプログラムの最初で実行しておかなければ、全く用をなさない場合がありますので注意してください。

計測器の応答がないとき

BREAKキーによりプログラムを停止させて、プログラムエラーを修正して再度RUN実行してください。

ウェイトルーチン

計測器にはデータの送受の準備完了までの待ち時間を設定する必要があります。

MZ-2000ではWAITステートメントがありませんので、FOR NEXT Loopで適当な待ち時間を設定してください。

相手側にPCT機能がないとき

PCTステートメントによって、他のコントローラにコントローラの権利を渡した後、そのコントローラが再度MZ-2000にコントローラの権利を戻す機能がないとき、ICLを実行すると相手のコントローラは初期設定され、MZ-2000にコントローラの権利が戻ります。

ただしMZ-2000はシステム・コントローラであることが必要です。

RS-232Cステートメント

Chapter 6

本章では、シリアルインタフェースカードMZ-8BI03を用いて、周辺機器をコントロールする場合のBASICステートメントに関する内容を含みます。

シリアルインタフェースの解説書（MZ80B用に作成された解説書ですが、MZ-2000用としても支障なく使用できます。）も参照ください。

6.1 ステートメント

6.1.1 RSMODE

書　式　RSMODE *a*, R*b*, T*c*, M*d*, RX*e*

a：チャンネル指定

a	チャンネル
A	Aチャンネル
B	Bチャンネル

b：受信キャラクタのビット数指定

c：送信キャラクタのビット数指定

b，*c*	ビット／キャラクタ
5	5
6	6
7	7
8	8

d：パリティビットの有無とストップビット数の指定

d	パリテイ	ストップビット
69	奇数	
70	無	1
71	偶数	
73	奇数	
74	無	1½
75	偶数	
77	奇数	
78	無	2
79	偶数	

e：受信可、不可の指定

e	受信
0	不可
1	可

機　能　上記パラメータにより各モードの設定を行います。

解　説
- すべてのパラメータを指定する必要はありませんが、受信可／不可の指定は各パラメータ指定の必ず最後におかなければなりません。
- モード設定パラメータ $a \sim e$ の指定は上記以外のものは使用しないこと。
- BASIC起動時には各チャンネル共、次のモードに設定されています。

b, c	8	キャラクタビット数8
d	79	偶数パリティ／ストップビット2
e	0	受信不可

例　10 RSMODE A, RX1………チャンネルAを受信可能とする。

6.1.2 RSO

書　式　RSO x $y$$

x：チャンネルの指定(AまたはB)

$y$$：ストリング変数で転送データを指定。

機　能　チャンネル x へ $y$$ で指定されるデータを送信します。

例　10 X$ = "Demonstration"

20 RSO B X$………チャンネルBへX$のデータを転送します。

6.1.3 RSI

書　式　RSI x $y$$

x：チャンネルの指定 (AまたはB)

$y$$：受信データを格納するストリング変数

機　能　チャンネル x からデータを受信し、ストリング変数 $y$$ に格納します。

例　10 RSMODE A, RX1 ………チャンネルAを受信可能とする。

20 RSI A B$ ……………………チャンネルAからデータを受信する。

30 PRINT B$

付　録　Appendix

付録には次のものが置かれています。

■ASCIIコード表………表A-1

■DISK BASICインタープリタMZ-2Z001エラーメッセージ表………表A-2

■メモリマップ………表A-3

■ディスクの取り扱い上の注意

A.1　ASCIIコード表

MZ-2000システムのASCIIコード表を次に示します。

10進	16進	キャラクタ	10進	16進	キャラクタ	10進	16進	キャラクタ	10進	16進	キャラクタ	10進	16進	キャラクタ
0	00	NULL	26	1A		52	34	4	78	4E	N	104	68	h
1	01	↓	27	1B		53	35	5	79	4F	O	105	69	i
2	02	↑	28	1C		54	36	6	80	50	P	106	6A	j
3	03	→	29	1D		55	37	7	81	51	Q	107	6B	k
4	04	←	30	1E	■	56	38	8	82	52	R	108	6C	l
5	05	HOME	31	1F	▦	57	39	9	83	53	S	109	6D	m
6	06	CLR	32	20		58	3A	:	84	54	T	110	6E	n
7	07	DEL	33	21	!	59	3B	;	85	55	U	111	6F	o
8	08	INST	34	22	"	60	3C	<	86	56	V	112	70	p
9	09	GRPH	35	23	#	61	3D	=	87	57	W	113	71	q
10	0A	SFT LOCK	36	24	$	62	3E	>	88	58	X	114	72	r
11	0B		37	25	%	63	3F	?	89	59	Y	115	73	s
12	0C	カナ	38	26	&	64	40	@	90	5A	Z	116	74	t
13	0D		39	27	'	65	41	A	91	5B	[	117	75	u
14	0E	L SCRIPT	40	28	(	66	42	B	92	5C	＼	118	76	v
15	0F	カナ CANCEL	41	29	)	67	43	C	93	5D	]	119	77	w
16	10		42	2A	*	68	44	D	94	5E	^	120	78	x
17	11		43	2B	+	69	45	E	95	5F	‾	121	79	y
18	12		44	2C	,	70	46	F	96	60	`	122	7A	z
19	13		45	2D	-	71	47	G	97	61	a	123	7B	{
20	14		46	2E	.	72	48	H	98	62	b	124	7C	|
21	15		47	2F	/	73	49	I	99	63	c	125	7D	}
22	16		48	30	0	74	4A	J	100	64	d	126	7E	~
23	17		49	31	1	75	4B	K	101	65	e	127	7F	↴
24	18		50	32	2	76	4C	L	102	66	f			
25	19		51	33	3	77	4D	M	103	67	g			

10進	16進	キャラクタ	10進	16進	キャラクタ	10進	16進	キャラクタ	10進	16進	キャラクタ	10進	16進	キャラクタ
128	80	┃┃┃	154	9 A	│	180	B4	エ	206	CE	ホ	232	E8	H
129	81	↓	155	9 B	─	181	B5	オ	207	CF	マ	233	E9	I
130	82	↑	156	9 C	┴	182	B6	カ	208	D0	ミ	234	EA	J
131	83	→	157	9 D	┬	183	B7	キ	209	D1	ム	235	EB	K
132	84	←	158	9 E	┤	184	B8	ク	210	D2	メ	236	EC	L
133	85	♠	159	9 F	├	185	B9	ケ	211	D3	モ	237	ED	M
134	86	♥	160	A0	V	186	BA	コ	212	D4	ヤ	238	EE	N
135	87	♦	161	A1	。	187	BB	サ	213	D5	ユ	239	EF	O
136	88	♣	162	A2	「	188	BC	シ	214	D6	ヨ	240	F0	Ø
137	89	┣	163	A3	」	189	BD	ス	215	D7	ラ	241	F1	1
138	8A	┫	164	A4	W	190	BE	セ	216	D8	リ	242	F2	2
139	8B	┳	165	A5	X	191	BF	ソ	217	D9	ル	243	F3	3
140	8C	┻	166	A6	ヲ	192	C0	タ	218	DA	レ	244	F4	4
141	8D	╋	167	A7	ァ	193	C1	チ	219	DB	ロ	245	F5	5
142	8E	╪	168	A8	ィ	194	C2	ツ	220	DC	ワ	246	F6	6
143	8F	╫	169	A9	ゥ	195	C3	テ	221	DD	ン	247	F7	7
144	90	═	170	AA	ェ	196	C4	ト	222	DE	゛	248	F8	8
145	91	¥	171	AB	ォ	197	C5	ナ	223	DF	゜	249	F9	9
146	92	U	172	AC	ャ	198	C6	ニ	224	E0	Z	250	FA	P
147	93	●	173	AD	ュ	199	C7	ヌ	225	E1	A	251	FB	Q
148	94	○	174	AE	ョ	200	C8	ネ	226	E2	B	252	FC	R
149	95	┐	175	AF	ッ	201	C9	ノ	227	E3	C	253	FD	S
150	96	┘	176	B0	Y	202	CA	ハ	228	E4	D	254	FE	T
151	97	┌	177	B1	ア	203	CB	ヒ	229	E5	E	255	FF	π
152	98	└	178	B2	イ	204	CC	フ	230	E6	F			
153	99	┼	179	B3	ウ	205	CD	ヘ	231	E7	G			

A.2 エラーメッセージ

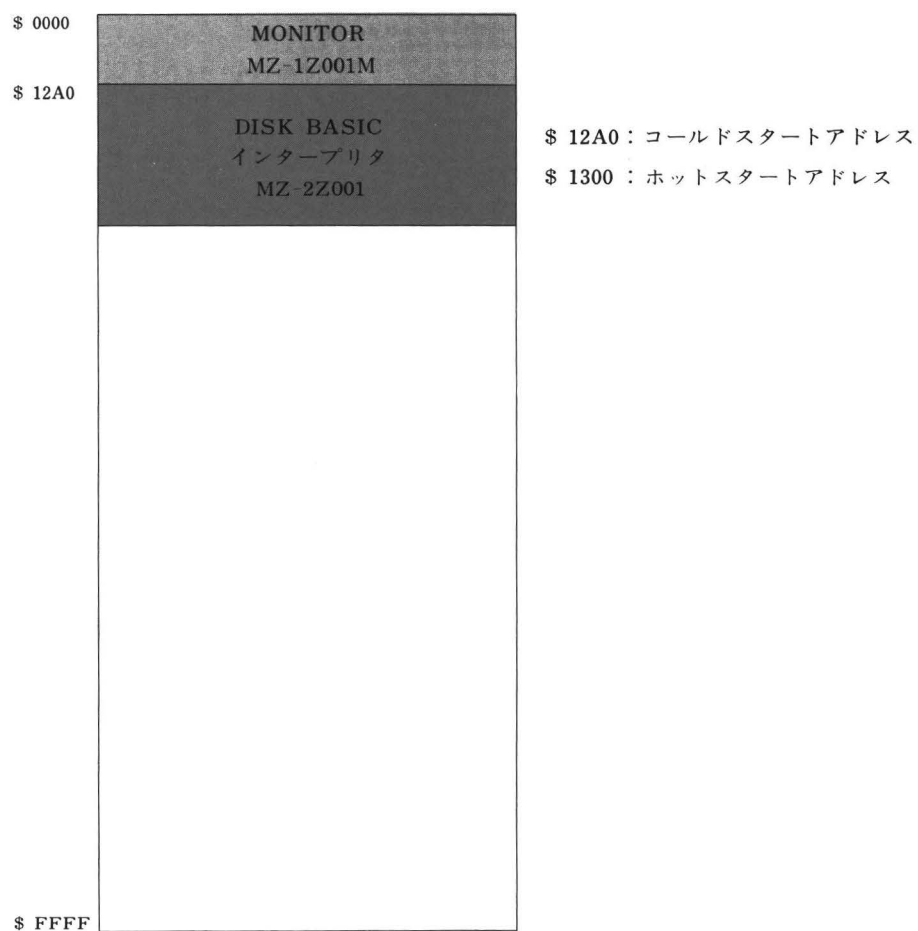
	エラー番号	エラーの内容
	1	文法上の誤り
	2	数値データが範囲外、演算結果がオーバフローした
	3	規定外の数値、変数が使われた
	4	データと変数の型が一致しない
	5	ストリングの長さが255文字を越えた
	6	メモリ容量不足となった
	7	同じ配列変数を前より大きく定義したか、未定義の配列変数を使用した
	8	1行の長さが制限を越えた
	9	
	10	GOSUB 文のネスティングが15を越えた
	11	FOR～NEXT 文のネスティングが15を越えた
	12	DEF FN 文による関数定義のネスティングが6を越えた
	13	対応する FOR 文のない NEXT 文が使われた
	14	対応する GOSUB 文のない RETURN 文が使われた
	15	定義されていない関数が使われた
	16	存在しない行番号を参照しようとした
	17	CONT 文によるプログラムの継続ができない
	18	BASIC インタープリタの管理エリア内への書き込み要求をした
	19	ダイレクトコマンドとステートメントを混同して使った
	20	RESUME が実行できない
	21	エラーが発生していないのに RESUME しようとした
	22	
	23	
	24	対応する DATA 文のない READ 文が使われた
	25	SWAP レベルが1を越えた
	26	
	27	
	28	
RS-232C	29	フレミングエラー
	30	オーバーランエラー
	31	パリティエラー
	32	データ転送不能（バッファが空でない）
	33	バッファのオーバーフロー
GP-IB	34	システムコントローラでないのにその命令を使用した
	35	カードまたはケーブルが接続されていない

	エラー番号	エラーの内容
GP-IB	36	3線ハンドシェークのNDACとNDTDが同時にHighレベル
	37	アドレスコードの設定エラー
	38	カード上でトーカ、リスナが同時にアクティブ
	39	リスナまたはトーカが能動でない
ディスク	40	存在しないファイルを参照した
	41	ディスクドライブのハード上のエラーが発生した
	42	すでに存在するファイル名を新たに登録しようとした
	43	すでにオープンされているファイルを更にオープンしようとした
	44	オープンされていないファイルを参照またはCLOSE、KILLした
	45	
	46	書き込み禁止ファイル
	47	
	48	
	49	
	50	ディスクドライブがレディ状態でない
	51	63を越すファイルを登録しようとした
	52	ボリュームナンバーエラー
	53	ディスク上のファイルスペースが無くなった
	54	イニシャライズされていないディスケット
	55	1つのBSDファイルのデータの総バイト数が64Kを越えた
	56	ディスクドライブコントロールルーチンでのデータエラー
	57	使用不能ディスク
ディスクまたはカセット	58	
	59	
	60	ファイル名エラー
	61	ファイルモードエラー
	62	
	63	カセットテープファイルデータ読み込みで、アウトオブファイルが起きた
	64	ロジカルナンバーエラー
プリンタ	65	プリンタが接続されていないか、OFF状態である
	66	プリンタにメカトラブルが起きた
	67	プリンタ用紙ぎれ
	68	
	69	
	70	チェックサムエラー

A.3　メモリ・マップ

アドレス	領域
$ 0000	MONITOR MZ-1Z001M
$ 12A0	DISK BASIC インタープリタ MZ-2Z001
$ FFFF	

$ 12A0：コールドスタートアドレス

$ 1300 ：ホットスタートアドレス

A.4　ディスクの取り扱い

ディスクのセットの仕方などディスクドライブの操作方法については、MZ-80BF等の取扱説明書をよくお読みください。

マスターディスクの取扱いには充分な注意が必要です。また、シャープオプションディスクはイニシャライズしておりませんので、ユーティリティプログラムによってイニシャライズしてから使用してください。

ディスクの取り扱い上の注意(使用時)

■　ヘッドウィンドウの箇所からディスクに指紋をつけたりすると、読み出し／書き込みが、永久に不能となります。ディスク表面は絶対に汚さないよう細心の注意が必要です。

■　保存温度条件：4℃～53℃（ディスク周囲温度）
53℃を起すとジャケットが変形します。直射日光に長時間さらしたり53℃を越す場所に置いたりしないでください。ディスクを使用する場合、エンベロープ（保護袋）に記載の温度範囲で使用するようにしてください。また保管場所と使用場所の環境条件が異なる場合がありますので、適用使用環境下にしばらく放置してから使用するように願います。

■　ドライブへの装着は、まっすぐに挿入し、止まる位置まで押し込み静かにフロントドアをしめてください。乱暴な取扱いはディスクをいためます。

■　曲げたり、折ったりしないでください。ジャケットが変形すると、読み出し／書き込みができなくなります。

■　インデックスラベルの書き込みは、ジャケットにはる前におこなってください。すでにはられたジャケットに書き込む場合は、鉛筆・ボールペンをさけサインペンなど先の軟らかいものを使用してください。

■　喫煙しながら、飲物を飲みながら、ディスクおよびディスクドライブを取り扱う場合は、灰や水滴で汚さないよう特に注意する必要があります。

ディスクの取り扱い上の注意(保管時)

■　磁石を近づけることは絶対にさけてください。マグネットリングや、マグネットネックレスなどもデータを消すことがありますからディスクを取り扱う時は、その使用をさけてください。その他の磁気を発生する機器に近づけることも危険です。たとえば、コンピュータのCRTディスプレイ、カセットレコーダ、家庭用テレビの近くなどでは磁気が発生していますので近づけないでください。

■　使用しない時はかならずエンベロープに入れてください。ドライブから取りだしたら、ただちにエンベロープに入れるように習慣づけてください。これによって、ディスクの取扱いの過ちによる大半の事故が防げます。特にマスターディスクは、使用する時にのみディスクドライブにセットし、使用しないときは大切に保管するようにしてください。エンベロープは、特殊な材質でできており、静電気・湿気等を防いでいますのでディスクは必ずエンベロープに収納してください。

■ 保管は、エンベロープに入れたものを、さらに収納箱に入れて垂直に立ててください。ななめに立てかけたり、ディスクがたわむような形で保管することはさけてください。
マスターディスクには、収納箱が付属しておりません。適当な箱を作って保管してください。

■ 紙バサミなどではさむことはさけてください。

■ ディスクの上に重いものを置かないでください。ディスクを無雑作に机の上に置くことはしないで、必ず所定の場所に保管してください。

シャープ株式会社

本　　　　　社　〒545 大阪市阿倍野区長池町22番22号
　　　　　　　　電話(06)621－1221(大代表)
産業機器事業本部　〒639-11 奈良県大和郡山市美濃庄町492番地
　　　　　　　　電話(07435)3－5521(大代表)
国内産機営業本部　〒162 東京都新宿区市谷八幡町8番地
　　　　　　　　電話(03)260-1161(大代表)

お客様ご相談窓口

札　幌 (011)642-4649	仙　台 (0222)88-9141	宇都宮 (0286)35-1155
東　京 (03)893-4649	金　沢 (0762)49-4649	名古屋 (052)322-4649
大　阪 (06)643-4649	広　島 (082)874-4649	高　松 (0878)33-4649
福　岡 (092)572-4649	沖　縄 (0988)62-2231	

シャープエンジニアリング株式会社

本社　〒114 東京都北区東田端2丁目13番17号　電話(03)800-1221(代表)

札幌支店	〒063 札幌市西区24軒1条7丁目3番17号	電話(011)641－4649
仙台支店	〒983 仙台市六丁目字本新田東2番地の1	電話(0222)88－9141
宇都宮支店	〒320 宇都宮市不動前4丁目2番41号	電話(0286)35－1155
東京支店	〒114 東京都北区東田端2丁目13番17号	電話 (03)800－1221
金沢支店	〒921 石川県石川郡野々市町字御経塚1096番地の1	電話(0762)49－4649
名古屋支店	〒454 名古屋市中川区山王3丁目5番5号	電話(052)332－2626
大阪支店	〒556 大阪市浪速区恵美須西1丁目2番9号	電話 (06)643－4649
広島支店	〒731-01 広島市安佐南区祇園町大字西原2249番地の1	電話(082)874－2281
高松支店	〒760 高松市木太町1861番地の3	電話(0878)33－4649
福岡支店	〒816 福岡市博多区井相田2丁目12番地の1	電話(092)572－4655
沖縄シャープ電機(株)	〒900 那覇市曙2丁目10番地の1	電話(0988)62－2231

シャープビジネス株式会社

本社　〒545 大阪市阿倍野区長池町22番22号　電話(06)621-1221(大代表)

札幌支店	〒063 札幌市西区24軒1条7丁目3番17号	電話(011)641－3631
仙台支店	〒983 仙台市六丁目字本新田東2番地の1	電話(0222)88－9151
東京支店	〒130 東京都墨田区石原2丁目12番3号	電話 (03)625－5111(代表)
千葉支店	〒280 千葉市南町1丁目5番20号	電話(0472)63－4043
横浜支店	〒235 横浜市磯子区中原1丁目2番23号	電話(045)751－3215
埼玉支店	〒330 大宮市宮原町2丁目107番地2号	電話(0486)63－5159
宇都宮支店	〒320 宇都宮市不動前4丁目2番41号	電話(0286)37－3576
新潟支店	〒950 新潟市上所中1丁目7番21号	電話(0252)83－1795
長野支店	〒380 長野市中御所2丁目11番3号	電話(0262)28－4618
名古屋支店	〒454 名古屋市中川区山王3丁目5番5号	電話(052)332－2631(代表)
金沢支店	〒921 石川県石川郡野々市町字御経塚1096番地の1	電話(0762)49－1240
大阪支店	〒556 大阪市浪速区恵美須西1丁目2番9号	電話 (06)643－3021(代表)
京都支店	〒601 京都市南区上鳥羽菅田町48番地	電話(075)661－7335
神戸支店	〒658 神戸市東灘区魚崎北町1丁目6番地18号	電話(078)452－8531
広島支店	〒731-01 広島市安佐南区祇園町大字西原2249番地の1	電話(082)874－4925
高松支店	〒760 高松市木太町1861番地の3	電話(0878)33－4255
福岡支店	〒816 福岡市博多区井相田2丁目12番地の1	電話(092)572－2611
沖縄支店	〒900 那覇市曙2丁目10番地の1	電話(0988)61－7360(代表)